UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Análisis, diseño e implementación de un sitio Web Departamental

Creación, modificación y almacenamiento de contenidos

Adolfo Miguel Catalán García-Manso

Director: Simon Pickin
16 Abril 2009

Agradecimientos

Con estas líneas quiero agradecer ante todo el esfuerzo y la paciencia de mis padres ya que sin ellos y sin su apoyo jamás hubiera llegado hasta aquí. Desde aquellas tardes que pasabais conmigo ayudandome a hacer la tarea hasta ahora que gracias a esa especial cualidad que teneis para espolearme y hacerme trabajar he terminado este proyecto.

También quiero agradecer a todos mis amigos y profesores ya que tanto unos como otros han hecho más llevadero mi paso por colegio y universidad bien apoyandome con su amistad bien inspirandome con sus palabras. En estos momentos al que más tengo que agradecer, por su puesto, es a mi director de proyecto, Simon, ese que en los últimos meses me ha dedicado seguramente más tiempo a mí que a sí mismo. Sin olvidarme de mi compañero de proyecto Carlos Holgado y su tutor Vicente gracias a los cuáles este proyecto comenzó a tener forma. También merece mención Carlos García, que junto con Simon establecieron las bases de este proyecto.

Finalmente, decir que aunque esta etapa llega a su fin, no será la última que haga ya que mi hambre de conocimientos sigue intacta y bien en este campo u en otros seguiré siendo estudiante por siempre.

Muchas gracias a todos.

Índice

1. Resumen

La aplicación Web en desarrollo tiene como finalidad la automatización y simplificación de las tareas de gestión del portal Web de cualquier departamento de la universidad (nos basamos en especial en el departamento de telemática pero bien podría ser aplicada a cualquier departamento de esta u otras universidades), permitiendo así mismo una mayor coherencia en los datos existentes en las páginas del departamento, por ejemplo entre las páginas de personal de IT, GAST y NETCOM.

Esta aplicación será desarrollada mediante los lenguajes de programación Java y Ruby, con una base de datos MySQL subyacente. Las páginas Web serán generadas de forma automática, o bajo la solicitud de un usuario con suficientes privilegios, a partir de un XML obtenido de la base de datos. Los datos que puedan ser representados mediante un estándar bibliográfico, usarán el estándar BibTeXML (sólo se aplicará este estándar para esta versión ya que está contemplada una mejora que permitirá incluir diferentes estándares), como es el caso de las publicaciones del departamento.

Se ha definido un sistema de usuarios de tipo incremental con diferentes privilegios de acceso y modificación de datos, compuesto por cuatro tipos de usuario, dependiendo del perfil que necesiten ejercer en el sistema. En los casos en los que personas con el mismo nivel necesiten distinto nivel de acceso a cierta parte de la base de datos (un profesor y el coordinador de una asignatura accediendo a ella), se comprobará por la característica de distinción, almacenada en la parte a la que intenta acceder.

La aplicación permitirá crear, modificar o borrar la información almacenada en la base de datos del departamento a través de formularios HTML. Entre la información disponible encontraremos desde los datos personales del personal docente hasta la información sobre las tesis o publicaciones que cuenten con algún miembro del departamento como autor o responsable. Además, permitirá notificar al interesado del cambio (creación, modificación, borrado), para los casos en los que no sea la persona interesada la que realiza el cambio en la Web (administrador de la aplicación).

Adicionalmente la aplicación permitirá gestionar otros formatos de entrada y salida para la información disponible. En esta versión sólo se contempla xml pero en mejoras futuras se podrían añadir otros formatos como podrían ser: Hojas Excel, documentos PDF e incluso la posible integración con sistemas externos como Universitas XXI. La biblioteca, como parte del Open Access Initiative, ha implantado un "Archivo Abierto Institucional", utilizando el software libre Dspace, para almacenar todo tipo de publicación y/o datos sobre publicaciones. Se esta estudiando la posibilidad de introducir los datos de publicaciones y de tesis en DSpace y utilizar los datos que exporta. Esta versión de la aplicación no toma en cuenta el uso de DSpace por parte de la biblioteca.

Actualmente, la aplicación es dependiente de los técnicos a la hora de crear los usuarios ya que son estos quien nos proporcionan el nombre y contraseña de usuario (personal administrativo completa los datos personales hasta cierto nivel y envía un correo electrónico a los técnicos para que generen los datos restantes que retornan al personal administrativo). Se

está considerando en cambio una posible ampliación que controle esto y permita que la aplicación controle todos los datos de un empleado.

En este proyecto se desarrolla la parte entrada de datos de una aplicación del tipo conocido como Sistema de Gestión de Contenidos (CMS, en sus siglas inglesas). La aplicación está diseñada para gestionar los sitios Web de un departamento universitario y de los distintos grupos de investigación que contiene este departamento, aunque se presta también a otros usos.

Para introducir la información del departamento en la base de datos los usuarios interactúan con la aplicación a través de formularios HTML. Mediante un sistema de perfiles y permisos, la aplicación descentraliza la tarea de introducción de datos, en la medida de lo práctico y lo conveniente: un empleado del departamento tiene permisos para crear, modificar o borrar los datos del sistema según su perfíl y sus atributos.

Por ejemplo, el coordinador de una asignatura tiene los permisos necesarios para modificar algunos de los datos que contiene el sistema sobre su asignatura, el responsable de un proyecto de investigación tiene los permisos para modificar los datos que contiene el sistema sobre el proyecto en cuestión, etc.

La parte de la aplicación desarrollada en este Proyecto Fin de Carrera sigue el patrón de diseño conocido como Modelo-Vista-Controlador (MVC), ampliamente usado en las aplicaciones Web de este tipo. Se ha implementado con el 'framework' para la construcción de aplicaciones Web "Ruby on Rails" y con el sistema de gestión de base de datos "MySQL".

La otra mitad de la aplicación, la parte de definición del modelo de datos y salida de datos (esto es, la generación automática de las páginas Web a partir de la información contenida en la base de datos) ha sido desarrollado por Carlos Holgado Molinillo en el marco de otro Proyecto Fin de Carrera (leído en octubre de 2008) titulado "Análisis, Diseño e Implementación de un Sitio Web Departamental: Recuperación y Publicación de Contenidos" dirigido por Simon Pickin y Vicente Luque Centeno del departamento de Ingeniería Telemática.

2. Glosario de Términos.

APC -	Alternative PHP Cache
API –	Application Programming Interface
AS –	Application Server
ASP -	Active Server Pages
B.B.D.D. –	Bases de Datos
BLOB -	Binary Large Objects
BSD -	Berkeley Software Distribution
CGI -	Common Gateway Interface
CLOB -	Character Large Object
CLU -	CLUsters
CMS -	Content Management Software
COC -	Convention Over Configuration
CPU –	Central Processing Unit
CSS -	Cascading Style Sheet
C.V -	Curriculum Vitae
DAO -	Data Access Object
DBMS -	Database Management System
DRY -	Don't Repeat Yourself
EJB -	Enterprise Java Beans
EL -	Expression Language
GAST -	Grupo de Aplicaciones y Servicios Telemáticos
GNU -	Gnu is Not Unix
GUI -	Graphic User Inteface
HTML -	Hypertext Mark-up Language
IDE -	Integrated Development Environment
IIOP -	Internet Inter-Orb Protocol
IoC -	Inversion of Control
ISAPI -	Internet Server Application Programming Interface
J2EE -	Java 2 Enterprise Edition
JDBC -	Java Database Connectivity
JMS -	Java Message Service
JMX -	Java Management eXtensions
JNDI -	Java Naming and Directory Interface

JTA -	Java Transaction Api
JTS -	Java Transaction Service
JSP –	Java Server Pages
JSR -	Java Specification Request
JVM -	Java Vitual Machine
LDAP -	Lightweight Directory Access Protocol
LGPL -	Lesser General Public License
LINQ -	Lenguaje Integrado de Consultas
LISP -	List Processing Language
MVC -	Modelo Vista Controlador
MVCC -	Multi-Version Concurrency Control
NETCOM -	Networks and Communication Seervices
ODBC -	Object DataBase Connectivity
O.O -	Object Oriented
PFC -	Proyecto Fin de Carrera
PHP -	PHP Hypertext Pre-processor
PHP CLI -	PHP Command Line Interface
PHP-GTK -	PHP Gimp Tool Kit
PHP-QT -	PHP Quasar Tecnologies
RHTML -	Ruby Hypertext Mark-up Language
RMI -	Remote Method Invocation
RoR -	Ruby on Rails
SA -	Servidor de Aplicaciones
SDK -	Software Development Kit
SCGI -	Simple Common Gateway Interface
SGBD -	Sistema de Gestión de Base de Datos
SOAP –	Simple Object Access Protocol
SQL -	Stuctured Query Language
SSIS -	SQL Server Integration Services
SSL -	Secure Socket Layer
S.S.O.O. -	Sistemas Operativos
TCL -	Tool Comman Language
TSL -	Test and Set Lock
URL -	Uniform Resource Locators
WAL -	Write Ahead Logging

XML -	eXtensible Markup Lenguaje
XML-RPC -	eXtensible Markup Lenguaje – Remote Procedure Calling
XSLT -	Extensible Stylesheet Language Transformation

3. Introducción.

Actualmente, el control del contenido de la Web de departamento de telemática de la universidad Carlos III se lleva de forma manual.

El procedimiento es el siguiente:

- Existe un Web master al que los usuarios envían las modificaciones que han surgido. Este envío se hace mediante e-mail generalmente.

- Los cambios se realizan sobre páginas html.

- La frecuencia de cambios no está fijada por lo que los cambios quedan pendientes hasta que exista un número suficiente de estos.

El los problemas que nos podemos encontrar son los siguientes:

- Las páginas del departamento no están actualizadas y es posible que incluso cuando se actualicen aún tengan datos no reales (se han podido producir cambios nuevos sobre los que se mantuvieron en cola para ser aplicados).

- También se plantea al problema de la actualización de varios sitios web distintos pero que comparten información. Esto actualmente se hace a mano lo que implica que se tiene que introducir la misma información por cada sitio donde se necesite. Por ejemplo:
  - Página Web del personal del departamento en inglés.
  - Página Web del personal del departamento en castellano.
  - Página Web del personal del grupo de investigación GAST en inglés.
  - Página Web del personal del grupo de investigación GAST en castellano.
  - Página Web del personal del grupo de investigación NETCOM en inglés.
  - Página Web del personal del grupo de investigación NETCOM en castellano.

A partir de esta situación y como proyecto de fin de carrera, se planteó la realización de una aplicación Web para el control de los datos de las páginas del departamento de telemática de una forma automática y sencilla.

Esta aplicación en principio permitiría al Web master cambiar todos los datos de las páginas Web y que estos cambios tuvieran efecto al día siguiente (Con esto nos referimos a la primera iteración del proyecto ya que en la siguiente iteración está planificado que sea parcial, es decir, que no hace falta regenerar todo el sitio Web sino que se hará a demanda). También permitirá a otros usuarios (secretarias, profesores y técnicos) la modificación de datos cumpliendo los permisos que se les otorguen según perfiles.

El proyecto fue dividido en dos vertientes que debido a su magnitud se asignó a diferentes alumnos:

- **Presentación de datos:** A partir de la información de b.b.d.d. se generan diariamente las páginas Web del departamento mediante un proceso nocturno. Esta vertiente sería creada mediante las tecnologías: Java, xslt y MySQL.

- **Aplicación de Control (creación / modificación / borrado) de datos:** Mediante formularios Web, el administrador puede controlar todos los datos de las páginas del departamento. Esta vertiente sería creada mediante Ruby on Rails, xml y MySQL.

Ambas vertientes debían de estar totalmente coordinadas por lo que los dos alumnos a los que se nos asignaron debíamos estar en contacto continuo para poder analizar la aplicación y determinar así el contenido y tipos de campos de la base de datos.

La aplicación de control de datos, que es de la que tratamos en este proyecto se trata de una aplicación Web que permite el acceso múltiple de usuarios filtrando por perfiles y que permite controlar en toda la información contenida en la b.b.d.d del departamento.

A partir de este momento siempre nos referiremos sólo a la aplicación de control de datos de la aplicación, que es la que ocupa este proyecto (Aparte del anexo A, que contiene el modelo de datos desarrollado mayoritariamente en el proyecto de Carlos Holgado Molinillo).

Durante la fase de análisis, se creó un foro Web, llamado "Nuevas páginas Web IT" (http://foros.it.uc3m.es/) en el que se permitía a los usuarios futuros de la aplicación discutir sobre las funcionalidades que esta tendría. En este sitio Web, se mostraban todos los casos de uso disponibles hasta la fecha (diagramas y templates).

El siguiente sitio web fue también creado (http://www.it.uc3m.es/spickin/webit/) para mostrar una línea a seguir para desarrollar la aplicación. Usuario: WebIT / Password: WebIT.

4. Análisis.

4.1 Objetivos

El objetivo principal del proyecto es crear una herramienta para el control y mantenimiento del contenido las páginas Web de los departamentos de la universidad, principalmente para el departamento de telemática. La herramienta debe ser sencilla, fácil de instalar y mantener en cualquier servidor de la universidad.

La aplicación que nos ocupa es de tipo CMS: Un Sistema de gestión de contenidos (*Content Management System*) es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos por parte de los participantes principalmente en páginas web.

Consiste en una interfaz que controla una base de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior que permite que estos contenidos sean visibles a todo el público.

De entre los diferentes tipos de CMS que existen este pertenece al tipo Portal: Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad. Ejemplos: PHP-Nuke, Postnuke, Joomla, Drupal, e-107, Plone, DotNetNuke,MS SharePoint.

4.1.1 Objetivos Formales.

El objetivo de la aplicación es crear un entorno Web por el cual se pueda controlar el contenido de las páginas de un departamento en concreto en tiempo real. El objetivo del PFC, que contempla sólo la vertiente de control de datos, es el de crear un entorno Web constituido por formularios html mediante el cual podemos crear, editar o borrar cualquier contenido de la Web.

- **Creación de un interfaz Web con la base de datos del departamento:** mediante formularios html y listado de datos se podrá operar con todas las entidades que conforman un departamento de la universidad. En el caso de este PFC, nos centraremos en el departamento de telemática pero bien podrá ser aplicado en otros departamentos con similar estructura.

- **Creación de un sistema de control de acceso a la información:** Hay definidos cuatro perfiles (administrador, administrativo, empleado y técnico). Cada uno cuenta con unas funcionalidades definidas y el control sobre los datos está establecido de forma gradual, es decir, el administrador tendrá disponibles todas las funcionalidades, el administrativo tendrá acceso sólo a parte de esas funcionalidades y así sucesivamente. También habrá funcionalidades que serán específicas de sólo un tipo de perfil.

- **Integración con "Presentación de Datos":** Dado que la aplicación fue dividida en dos apartados debido a su magnitud, la aplicación de control de datos debe adaptar los datos que se crean al formato que tiene definida la aplicación de salida.

- **Sistema multiplataforma:** La aplicación de control de datos no deberá ser dependiente del sistema operativo de los servidores.

- **Sistema multidioma:** Dado que el uso de la aplicación esta contemplado sólo para personal de la universidad, se extiende que se utilizará sólo el castellano para esta. No ocurrirá lo mismo para los datos controlados por la aplicación, que abarcan el castellano y el inglés como mínimo.

4.1.2 Objetivos Operacionales.

La aplicación debe ser simple para el usuario. Se controlarán todos los posibles pasos del usuario en la medida de lo posible para evitar errores de operaciones. Así mismo el acceso y modificación de los datos estará restringido según perfiles.

1.- El interfaz con el usuario debe ser claro e intuitivo, que conduzca todas las posibles acciones que pueda realizar el usuario.

2.- Todos los formularios Web de los que está compuesta la aplicación siguen el mismo formato. Sin estilos aplicados, como fue especificado en los requerimientos, para que de esta manera se pueda aplicar el estilo deseado cuado se utilice la aplicación. Sin embargo se tratará de aplicar un estilo en la demo de la aplicación.

3.- Tratamos que en todo momento la aplicación cumpla la estructura Modelo–Vista–Controlador para que sean más fáciles posibles modificaciones y/o ampliaciones de esta.

4.- Se comentará el código de la aplicación para que éste sea más fácil de comprender por otros programadores.

5.- La aplicación será independiente de la plataforma en la que se ejecute.

6.- La aplicación no consumirá más de los recursos especificados en los requerimientos y siempre tratará de hacer uso de software libre.

4.2 Definición del Problema.

Se quiere crear una aplicación Web que lleve el control del contenido de la Web de departamento de telemática de la universidad Carlos III, que en la actualidad se lleva de forma manual por una única persona:

El Web master de la página del departamento de telemática recibe correos de diferentes fuentes (secretarias, empleados, técnicos, etc.) en los que se indican cambios en la información a mostrar el la página Web del departamento.

Las páginas del departamento son ficheros HTML por lo que aplicar cualquier cambio sobre ellos no resulta trivial sino que en ocasiones resulta tedioso.

Los cambios son esporádicos por los que se espera a recibir un número aceptable de ellos antes de proceder a la modificación de los HTML.

Al ser una única persona el Web master del departamento, en ocasiones los cambios dependen de la disponibilidad de esta persona.

Por consiguiente, la Web del departamento no esta actualizada en tiempo real. Con ello se pierden datos de posible relevancia, por ejemplo cambios, altas y bajas de personal, cambios de despachos, teléfonos, nueva información de los empleados, PFC, doctorados, etc.

4.3 Informe de necesidades.

A la vista del problema anterior, se busca crear una plataforma Web que gestione de forma fácil y sencilla, incluso por una única persona, los datos contenidos en la Web del departamento.

La aplicación Web estará compuesta por formularios HTML mediante los cuales se controlará la información incluida en la b.b.d.d.

Se controlará y restringirá el acceso a la información de la b.b.d.d. estableciendo niveles de seguridad por perfiles.

Sistema de Usuarios: Las funcionalidades de los usuarios que acceden a la aplicación vienen determinada por perfiles:

Administrador:

- Se trata del Web master.
- Tiene todos los derechos sobre la aplicación y b.b.d.d.
- Crea/edita/borra Departamentos.
- Crea/edita/borra Publicaciones.
- Crea/edita/borra Asignaturas.
- Crea/edita/borra Anuncios (Eventos).
- Crea/edita/borra Grupos.
- Crea/edita/borra Titulaciones.
- Crea/edita/borra Líneas de Investigación.
- Crea/edita/borra Proyectos de Investigación.
- Crea/edita/borra Tesis.
- Crea/edita/borra Empleados.
- Crea/edita/borra cualquier elemento necesario para definir los anteriores.

Administrativo:

- Se aplica este perfil a cualquier empleado de la universidad con funciones administrativas.
- En esta versión de la aplicación se ha dotado al administrativo de las mismas funciones que el administrador, tan sólo capando algunas funciones (caso creación/ edición / borrado de Empleado).

Empleado:

- Todos los empleados de la universidad inicialmente tienen este perfil.
- Sólo tienen capacidad de modificación de los datos de los que son directamente responsables.

Técnico:

- Sólo desempeñan la función de completar los datos de un nuevo empleado.

Todos estos perfiles interactúan entre sí a la hora de crear un nuevo Empleado en la aplicación.

Este es el caso principal de la aplicación. A partir de la creación de un Empleado y establecidos sus datos, ese mismo Empleado puede realizar modificaciones en la b.b.d.d. dependiendo del perfil que se le haya adjudicado.

4.4 Restricciones.

A continuación pasaremos a detallar las restricciones que podemos encontrar entorno a la aplicación y que finalmente condicionarán las decisiones tomadas a la hora del análisis.

4.4.1 Factores externos al proyecto.

Se pueden dar situaciones ajenas al proyecto que pueden interferir de forma notable en el desarrollo de este:

- Disponibilidad tanto de proyectistas como de directores del proyecto a la hora del seguimiento del proyecto. Ya sea mediante reuniones periódicas, e-mails o tutorías. Se puede dar el caso que alguno de los dos mencionados no este disponible en el seguimiento del proyecto lo que implicaría retrasos en el mismo, toma de decisiones por cuenta propia. Con la consiguiente posibilidad de errores en esta toma de decisiones. Como nota personal a este punto, hubo que coordinar muy cuidadosamente el proceso de análisis y diseño del proyecto, ya que durante este tiempo no estuve disponible para reuniones personales debido a una ausencia de un año por estudios/trabajo en China (Shanghai) pero se continuó con el desarrollo de estas fases del proyecto vía emails.

- Limitación de recursos: Dependeremos en todo momento bien de equipos de la universidad o bien de equipos propios. En el primer caso estaremos condicionados por la disponibilidad de equipos, horarios universitarios, etc. Mientras que en el segundo de los casos, puede darse el caso de que no dispongamos el software o el hardware necesarios para llevar a cabo el desarrollo del proyecto.

- Limitaciones de tiempo: Como se ha mencionado antes, dependeremos de los horarios universitarios respecto a consultas a directores de proyecto y equipos. También estaremos condicionados a los límites de tiempo impuestos en la entrega de proyectos de fin de carrera. En este caso se contempla un mínimo de 4 meses y un máximo de 2 años en el desarrollo del proyecto.

- Inicialmente no se han previsto limitaciones económicas para el proyecto al tratarse del desarrollo de una aplicación interna para la universidad. Sin embargo se ha considerado que esta aplicación consumirá tan sólo los sistemas necesarios para su funcionamiento.

4.4.2 Factores propios del proyecto.

En este punto se deberán localizar todas las restricciones propias del proyecto que serán las condicionantes finales a la hora de decantarnos por un tipo de arquitectura o tecnología en el desarrollo de éste.

Esto es tarea de los proyectistas, que mediante reuniones con los directores de proyecto y usuarios de la aplicación desarrollarán un listado de requisitos que debe cumplir la aplicación. Este documento es vital en el desarrollo de la aplicación ya que enmarca el proyecto a realizar por los proyectistas, justifica las decisiones de la solución final encontrada y establece la manera de realizar la aplicación así como las acciones que puede realizar esta.

Pudimos comprobar que existen aplicaciones de software libre de bastante uso en el mercado, con lo que posiblemente se hubiese podido encontrar alguna aplicación con suficientes ventajas y pocos inconvenientes que hubiese facilitado desarrollar el sistema en estudio, como podrían ser las aplicaciones Joomla o Drupal, dos de las de uso más extendido dentro del software libre. Sin embargo, se descartartó el uso de estas aplicaciones por dos razones principales: empobrecían el aprendizaje del alumno, que había sido definido como objetivo principal del proyecto, y limitaban la extensibilidad del sistema a la hora de generar nuestra web.

Para la solución de una alternativa adecuada debemos de seleccionar diferentes arquitecturas y tecnologías a usar:

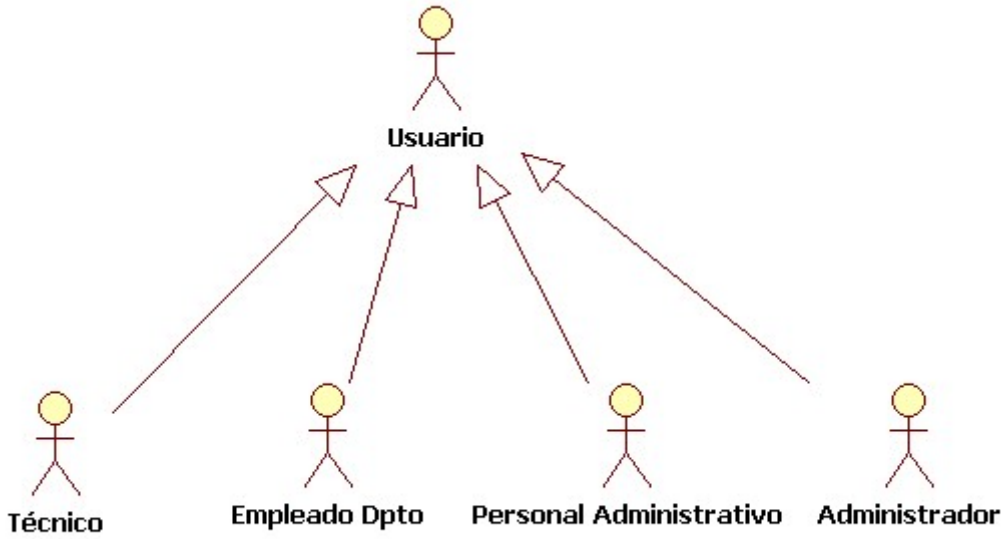
1. **Arquitectura o entorno de ejecución.**
   - J2EE (JAVA).
   - Ruby On Rails.
   - PHP.
   - Python.

2. **Servidor de aplicaciones.**
   - JBOSS.
   - BEA Weblogic.
   - WEBrick Server.
   - Mongrel Server.
   - Light TPD.

3. **Framework Web (Modelo-Vista-Controlador).**
   - Struts (JAVA).
   - Spring (JAVA).
   - Ruby On Rails.

4. **Base de Datos**
   - SQL Server.
   - MySQL.
   - PostgreSQL.

4.5 Sistema de perfiles.



Disponemos de cuatro tipos de usuario diferenciados por sus perfiles dentro de la aplicación. Esta es sólo una definición preliminar que irá avanzando con futuras implementaciones de la aplicación. A continuación presentamos los perfiles:

- **ADMINISTRADOR:**

Es el perfil con más privilegios de la aplicación. Tiene permisos sobre todos los objetos y le está permitido realizar todas las acciones que crea conveniente sobre ellos. Inicialmente se pensó este perfil para ser asignado al Web Master de la aplicación. De esta forma podría realizar las actividades de actualización de la Web del departamento de Telemática salvo que con esta aplicación y este perfil, se le permitiría modificar el contenido de las páginas Web del departamento casi en tiempo real (demora de 24h máximo). Evitando así la tarea de modificar directamente páginas HTML.

- **PERSONAL ADMINISTRATIVO:**

Este perfil cuenta inicialmente con unos permisos similares a los del administrador. Tan sólo difiere, en esta versión, en el caso de uso de EMPLEADO, en el que se le ha dotado la capacidad de modificar sólo parte de los datos del empleado. Se pensó así ya que se quería dotar a otro tipo de usuario de unos permisos similares a los del Web master para distribuir así la carga de trabajo. De esta forma y para datos relacionados con la administración, se relevará al administrador de la tarea de introducir estos datos. Se otorgará este perfil a personal de secretaría del departamento. I

- **EMPLEADO DEPARTAMENTO:**

Este será el perfil más común de los usuarios de la aplicación. Se otorgará a los profesores del departamento principalmente. Cuenta con una visión más reducida de la aplicación y sus movimientos están restringidos sólo a aquellos objetos de los que es propietario.

- **TÉCNICO:**

En esta versión, este será el perfil con menos capacidades. Sólo cuenta con la tarea de modificar los datos de EMPLEADO.

4.6 Casos de Uso.

A continuación se detallarán cada uno de los sistemas de casos de uso de los que está dotada la aplicación indicando que perfiles pueden participar y de qué operaciones están compuestas.

Este es un resumen de las adaptaciones de los casos de uso reales. Sólo son válidas para esta iteración de la aplicación. Los casos de uso completos están anotados y detallados en el anexo B de esta memoria. Se hace notar al lector de esta memoria que para poder omprender mejor el funcionamiento de esta aplicación debe referirse primero al anexo nombrado.

- **Gestión de Asignaturas 1: (Anexo B 10.2.1.2)**
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.
  - LOGIN: Para gestionar las asignaturas es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR ASIGNATURA: Se crea una nueva asignatura. Se mostrará un formulario con todos los campos a rellenar por el usuario. En caso de error, se notificará al usuario el campo erróneo y se le pedirá que vuelva a introducir los datos. Implica la posibilidad de asignar empleados que imparten la asignatura. Una vez creada, se notifica este hecho al administrador, administrativo y a todos los empleados que hayan sido asignados.

  - MODIFICAR ASIGNATURA: Se modifica una asignatura ya existente. Se selecciona de un listado de asignaturas y se muestra un formulario con todos los datos correspondientes a la asignatura permitiendo la modificación de todos ellos. También se mostrarán listados de los elementos que dependen de una asignatura, como es el caso de empleados que imparten la asignatura y se permitirá que estos también sean modificados en la forma que se desee. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR ASIGNATURA: Se elimina una asignatura definitivamente de la aplicación. Se eliminarán también todas las relaciones de esta asignatura. No se guarda un histórico de asignaturas.

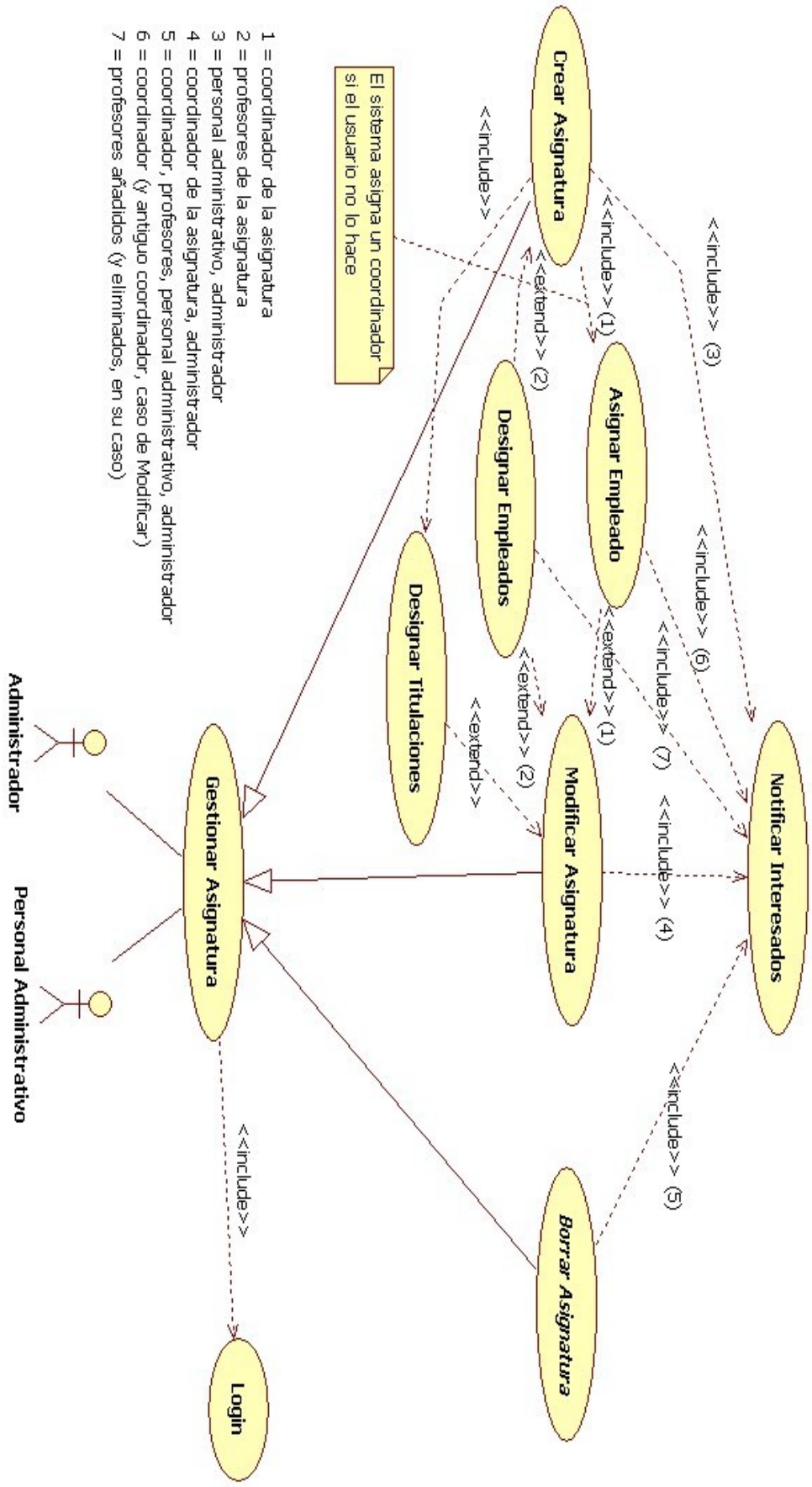
  - ASIGNAR EMPLEADO: Se muestra un listado de empleado que no están impartiendo la asignatura para que el usuario los incluya en el listado de profesores que imparten la asignatura.

- NOTIFICAR INTERESADOS: Se enviará un correo electrónico tanto al administrador como a todos aquellos empleados que tengan alguna relación con la asignatura en concreto independientemente del perfil que desempeñen.

Diferencias con el modelo implementado: Se ha modificado el "workflow" de la aplicación para que cuando se cree una asignatura, se pida al usuario que indique qué empleados son los que impartirán la asignatura y a qué titulación/es pertenece la asignatura creada. Inicialmente se pedían todos estos datos en el formulario inicial de creación de asignaturas pero resultaba demasiado complicado por lo que se dividió la acción en varios formularios (creación de asignatura, asignar empleados, asignar las titulaciones. Para ello se ha tenido que desarrollar estos casos de uso).

Ya que estos datos son obligatorios, no se le permite al usuario abandonar estos casos de uso hasta que los haya completado.

Igualemente, cuando se modifique una asignatura, se tendrán en cuenta todas las condiciones en relación al número de empleados que imparten la asignatura y a cuantas titulaciones debe pertenecer como mínimo. Tampoco se le permitirá al usuario realizar acciones que contradigan estas condiciones.

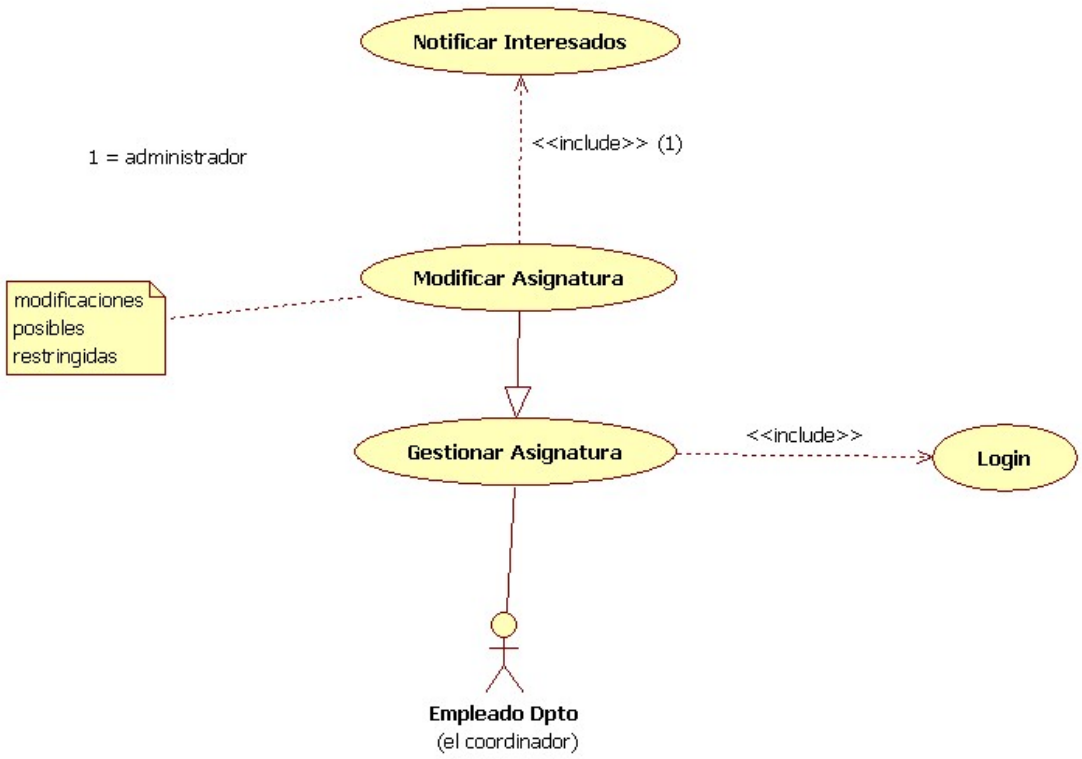

Crear Asignatura
Asignar Empleado
Designar Empleados
Designar Titulaciones
Modificar Asignatura
Notificar Interesados
Borrar Asignatura
Login
Gestionar Asignatura
Administrador
Personal Administrativo
El sistema asigna un coordinador si el usuario no lo hace
1 = coordinador de la asignatura
2 = profesores de la asignatura
3 = personal administrativo, administrador
4 = coordinador de la asignatura, administrador
5 = coordinador, profesores, personal administrativo, administrador
6 = coordinador (y antiguo coordinador, caso de Modificar)
7 = profesores añadidos (y eliminados, en su caso)
<<include>>
<<include>> (1)
<<include>> (3)
<<include>> (6)
<<include>> (7)
<<include>> (4)
<<include>> (5)
<<extend>> (2)
<<extend>> (1)
<<extend>> (2)
<<extend>>

- **Gestión de Asignaturas 2: (Anexo B 10.2.1.2)**

  Este sistema se limita al perfil de EMPLEADO.

  - LOGIN: Para gestionar las asignaturas es necesario estar logado e identificado por el perfil mencionado anteriormente.

  - MODIFICAR ASIGNATURA: Se modifica una asignatura ya existente. Se selecciona de un listado de asignaturas y se muestra un formulario con todos los datos correspondientes a la asignatura permitiendo la modificación de todos ellos. También se mostrarán listados de los elementos que dependen de una asignatura, como es el caso de empleados que imparten la asignatura y se permitirá que estos también sean modificados en la forma que se desee. En este caso el empleado deberá ser coordinador de la asignatura. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Para que el usuario EMPLEADO pueda modificar una asignatura debe ser coordinador de la asignatura.



- **Gestión de Campus: (Anexo B 10.2.1.8)**

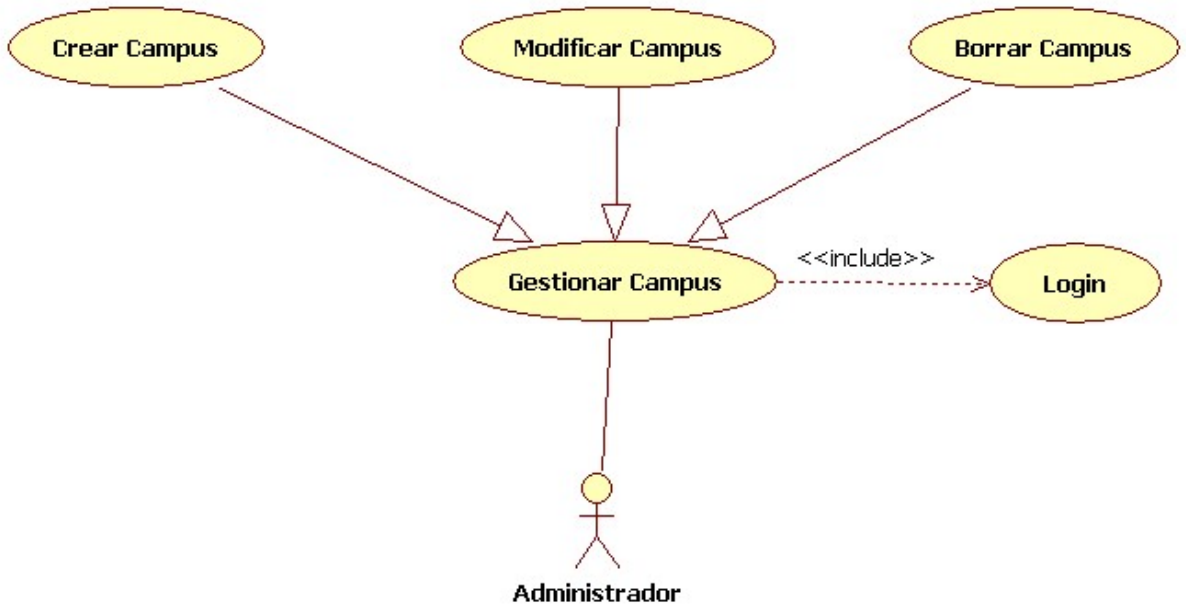
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Campus es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR CAMPUS: Se crea un nuevo Campus a gestionar por la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Campus. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR CAMPUS: Se muestra un listado con los Campus disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Campus, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR CAMPUS: Se elimina el Campus definitivamente de la aplicación. Se eliminarán también todas las relaciones de este campus, es decir, despachos. No se guarda un histórico.



- **Gestión de Cargos: (Anexo B 10.2.1.11)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

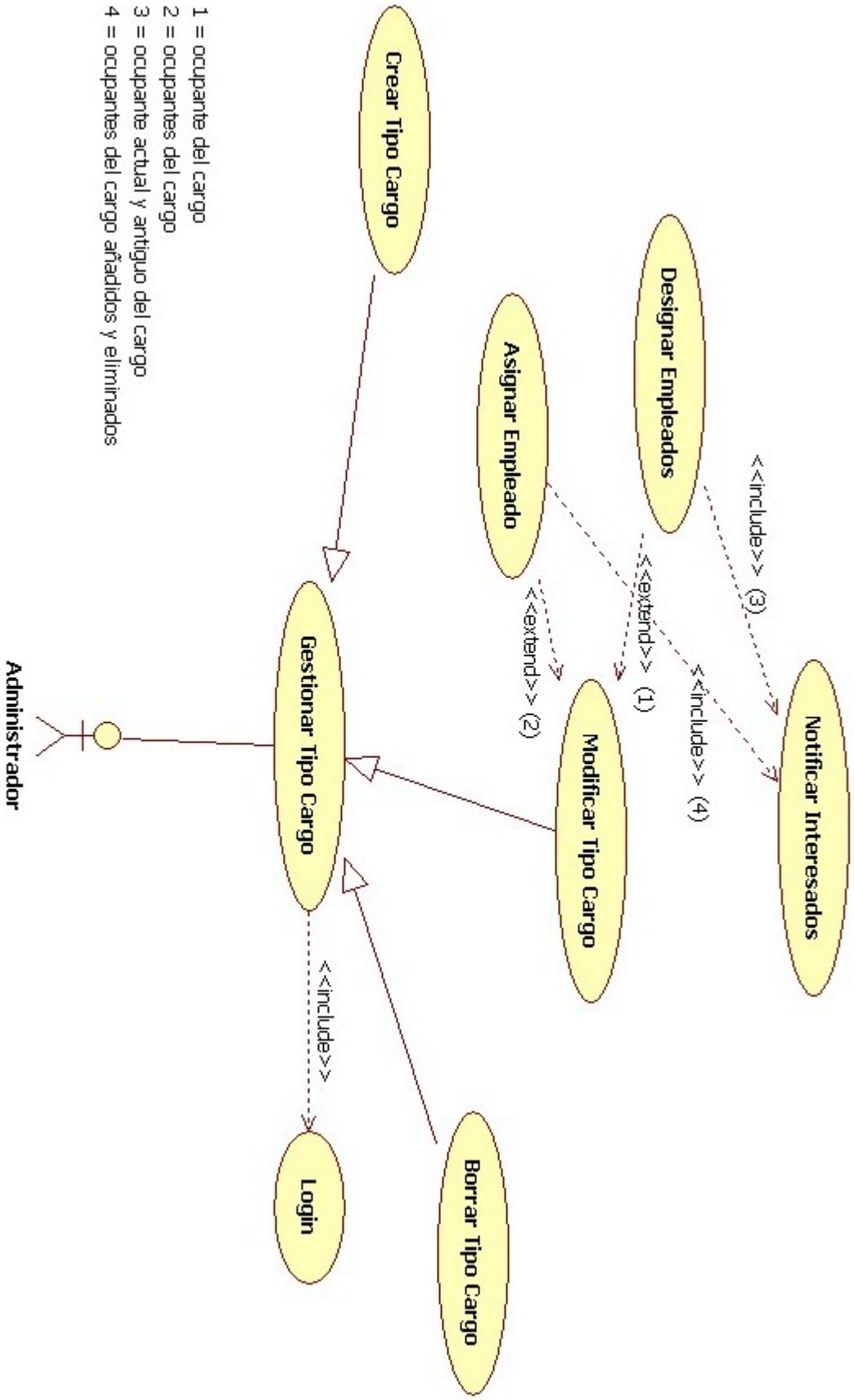
  - LOGIN: Para gestionar los Cargos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR CARGO: Se crea un nuevo Tipo de Cargo para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Tipo de Cargo. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR CARGO: Se muestra un listado con los Tipos de Cargo disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Tipo de Cargo, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Cualquier cambio realizado afectará a todos los empleados que posean este cargo.

  - BORRAR CARGO: Se elimina el Tipo de Cargo definitivamente de la aplicación. Antes de poder eliminar un Tipo de Cargo se pedirá al usuario que reasigne el cargo de todos los empleados que poseen el Tipo de Cargo a eliminar. En caso de no ser así no se eliminará. No se guarda un histórico.

    Diferencias con el modelo implementado: En el formulario para modificar cargos no se le permite modificar el cargo de los empleados. Se ha implementado de esta forma considerando un caso fututo en el que varios empleados puedan tener el mismo cargo. De esta forma, si se requiere modificar el cargo de un empleado, se deberá acceder al formulario de modificación de ese empleado y realizar el cambio (ya que el cargo de un empleado corresponde a los datos del empleado). Se reserva el formulario de modificación de cargos para modificar campos como la descripción del cargo.


1 = ocupante del cargo
2 = ocupantes del cargo
3 = ocupante actual y antiguo del cargo
4 = ocupantes del cargo añadidos y eliminados
Crear Tipo Cargo
Designar Empleados
Asignar Empleado
Notificar Interesados
<<include>> (3)
<<extend>> (1)
<<extend>> (2)
<<include>> (4)
Gestionar Tipo Cargo
Modificar Tipo Cargo
Administrador
<<include>>
Login
Borrar Tipo Cargo

- **Gestión de Tipos de Evento: (Anexo B 10.2.1.11)**

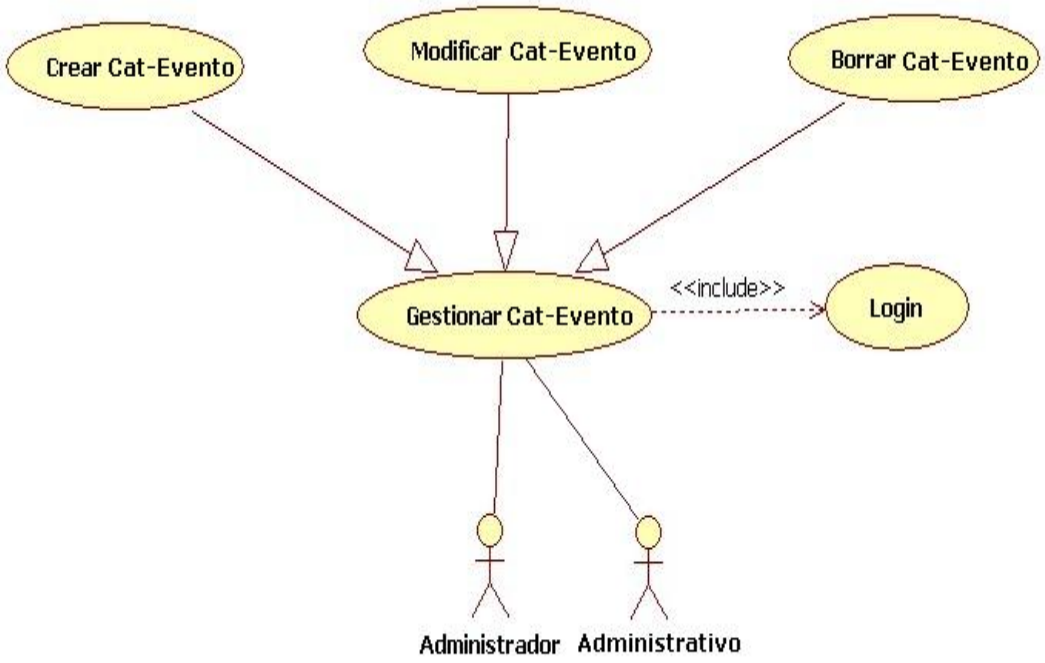
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Tipos de Eventos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR TIPO DE EVENTO: Se crea un nuevo Tipo de Evento para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Tipo de Evento. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR TIPO DE EVENTO: Se muestra un listado con los Tipos de Evento disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Tipo de Evento, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Cualquier cambio realizado afectará a todos los eventos que posean este tipo.

  - BORRAR TIPO DE EVENTO: Se elimina el Tipo de Evento definitivamente de la aplicación. Antes de poder eliminar un Tipo de Evento se pedirá al usuario que reasigne el tipo de todos los eventos que poseen el Tipo de Evento a eliminar. En caso de no ser así no se eliminará. No se guarda un histórico.



- **Gestión de Despachos: (Anexo B 10.2.1.8)**

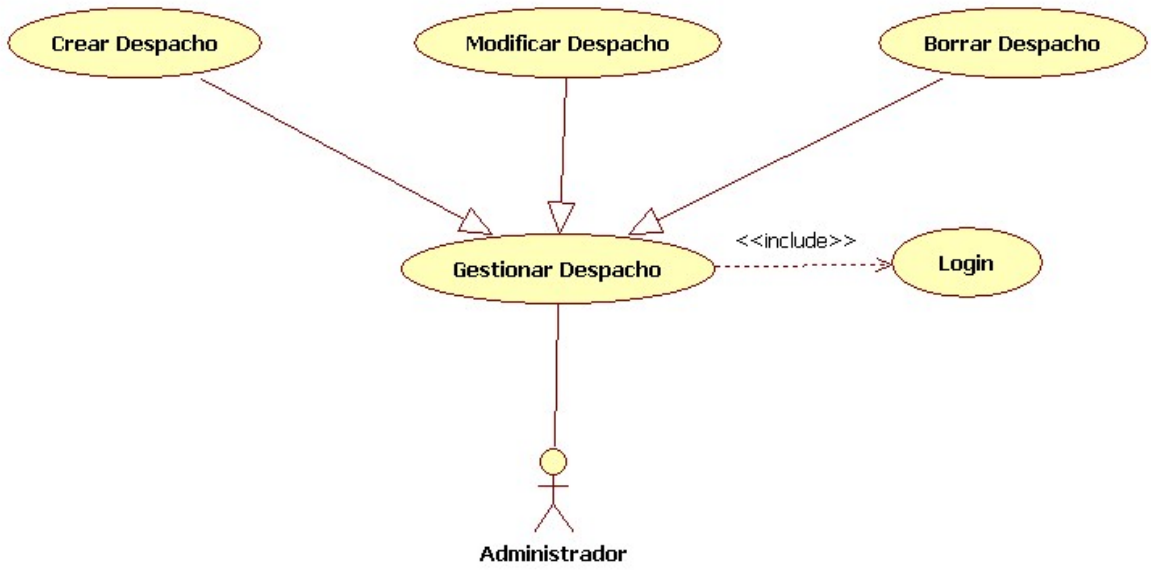
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Despachos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR DESPACHO: Se crea un nuevo Despacho para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Despacho. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR DESPACHO: Se muestra un listado con los Despachos disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos del Despacho, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR DESPACHO: Se elimina el Despacho definitivamente de la aplicación. Antes de poder eliminar un Despacho se pedirá al usuario que reasigne el despacho de todos los empleados que estén asociados a éste. En caso de no ser así, no se eliminará el Despacho. No se guarda un histórico.



- **Gestión de Empleados 1: (Anexo B 10.2.1.1)**

Para asegurar la coherencia entre las tres bases de datos del departamento IT que contienen información sobre los empleados – la de la presente aplicación (base WebIT), la de secretaría (base Access) y la de los técnicos (base LDAP) – se propone el siguiente proceso para el alta de un empleado:

1. El personal adminstrativo creará el nuevo empleado mediante el caso de uso Crear Empleado, sin que pueda rellenar los "datos de cuenta"[1]; el sistema generará un valor único para el atributo login_local (puesto que no puede ser nulo). Por otro lado, el front-end de la base Access se modificará con el fin de evitar la introducción por teclado de datos antes de importar los datos de la base WebIT.
2. El técnico recibirá un aviso por e-mail y seguidamente completará los datos del nuevo empleado introduciendo los "datos de cuenta" del nuevo empleado en la base WebIT (actualmente, por teclado mediante el caso de uso Modificar Empleado), además de introducir la parte correspondiente de los datos del nuevo empleado en la base LDAP.
3. El personal adminstrativo recibirá un aviso por e-mail y seguidamente, usando el front-end modificado mencionado antes, recuperará de la base WebIT mediante consultas SQL los datos del nuevo empleado que son comunes entre las bases Access y WebIT, los completará por teclado con los datos exclusivos de la base Access y los almacenará en esta última base.
4. Cualquier modificación posterior de datos comunes Access-WebIT se introducirá primero en la base WebIT, mediante Modificar Empleado, y se importará luego por el front-end de la base Access.

Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

- LOGIN: Para gestionar los Empleados es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

- CREAR EMPLEADO: Se crea un nuevo Empleado a gestionar por la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Empleado de la aplicación. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

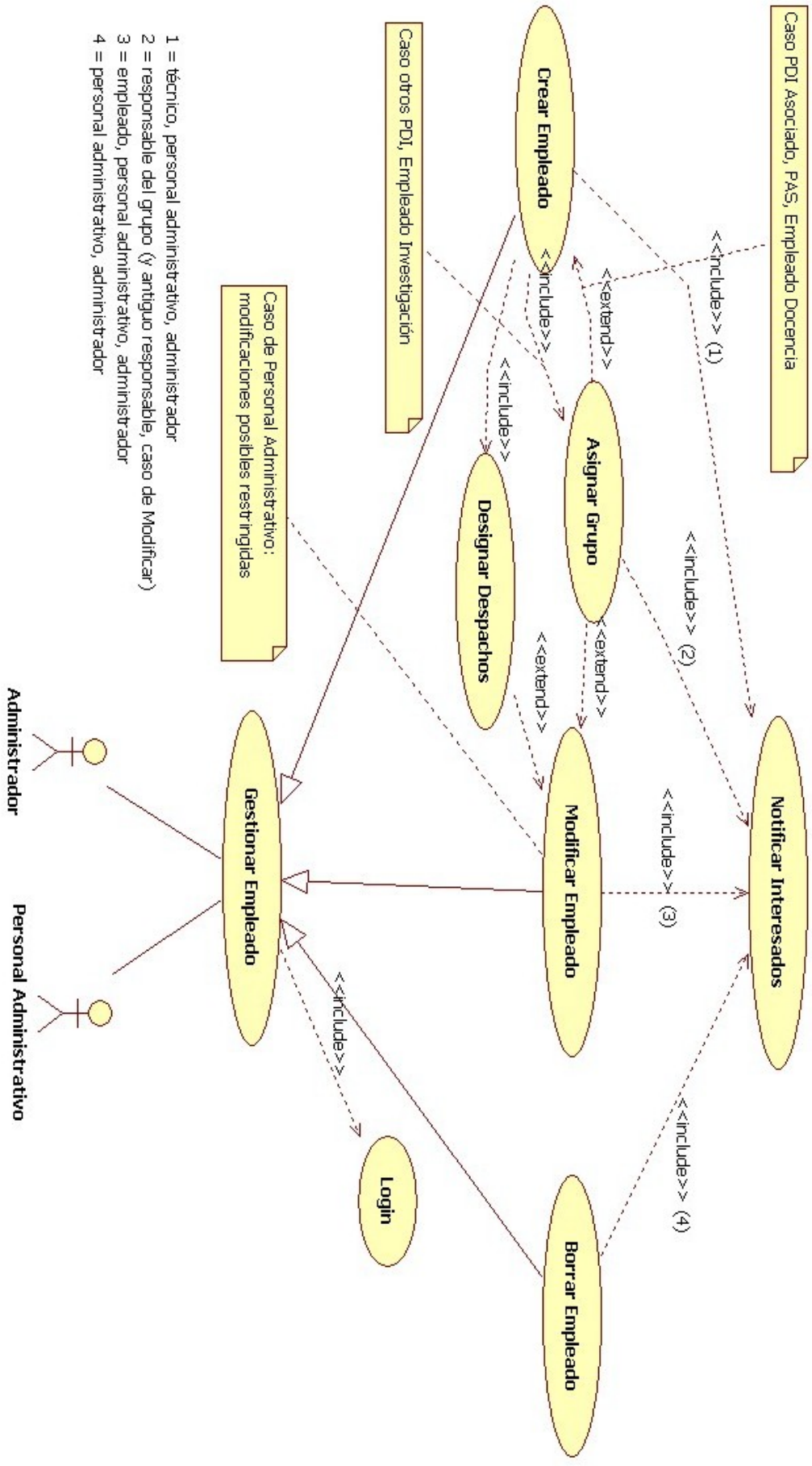
- MODIFICAR EMPLEADO: Se muestra un listado con todos los Empleados disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos del Empleado (el perfil ADMINISTRATIVO tendrá limitados los campos a modificar), permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

- BORRAR EMPLEADO: Se elimina el Empleado definitivamente de la aplicación. Se eliminarán también todas las relaciones de este Empleado. Para eliminar un Empleado se deben modificar primero todas las relaciones que este tenga con otras entidades de la aplicación, es decir, si es autor, director, responsable, etc. Si no se modifican estas relaciones el Empleado no será eliminado. No se guarda un histórico.

[1] Los "datos de cuenta" de un empleado se definen actualmente como los atributos siguientes: login_local, e-mail y URL_pagina.

- ASIGNAR GRUPO: Tanto en la creación como en la modificación de Empleados, le es permitido al usuario modificar el grupo al que pertenece un Empleado. Todo empleado debe pertenecer a un grupo por lo que asignar un grupo se hace obligatoriamente durante la creación y se permite el cambio de grupo durante la modificación de Empleado.

- NOTIFICAR INTERESADOS: Cuando se realiza una operación sobre un Empleado se notifica automáticamente y mediante correo electrónico tanto al ADMINISTRADOR de la aplicación como al Empleado que ha sufrido la operación.


Caso PDI Asociado, PAS, Empleado Docencia
Caso otros PDI, Empleado Investigación
Caso de Personal Administrativo: modificaciones posibles restringidas
1 = técnico, personal administrativo, administrador
2 = responsable del grupo (y antiguo responsable, caso de Modificar)
3 = empleado, personal administrativo, administrador
4 = personal administrativo, administrador
Crear Empleado
Asignar Grupo
Designar Despachos
Notificar Interesados
Modificar Empleado
Gestionar Empleado
Borrar Empleado
Login
Administrador
Personal Administrativo
<<include>> (1)
<<include>> (2)
<<include>> (3)
<<include>> (4)
<<extend>>

- **Gestión de Empleados 2: (Anexo B 10.2.1.1)**

  Este sistema se limita a los perfiles de EMPLEADO y TÉCNICO.

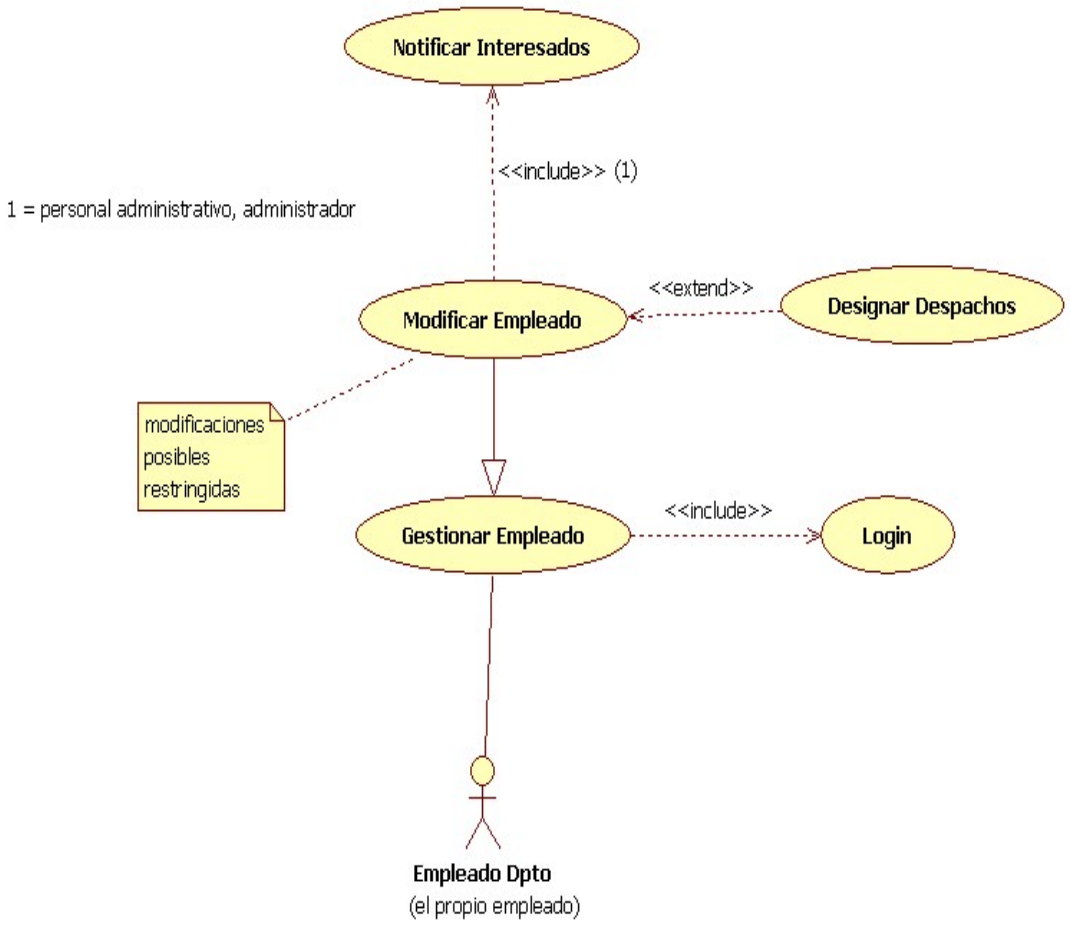
  - LOGIN: Para gestionar un Empleado es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - MODIFICAR EMPLEADO: Se muestran dos perspectivas dependiendo del perfil del usuario:

    1.- EMPLEADO: Tan sólo se muestra el empleado que se corresponde con el usuario. Permitiéndole modificar sólo los datos personales. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

    2.- TÉCNICO: Se muestra un listado con todos los empleados disponibles permitiendo modificar sólo los datos referentes al nombre de usuario y cuenta dentro de la universidad.

  - NOTIFICAR INTERESADOS: Cuando se realiza una acción sobre un Empleado, en todo caso se comunicará al ADMINISTRADOR de la aplicación y al EMPLEADO afectado mediante correo electrónico.



- **Gestión de Eventos: (Anexo B 10.2.1.7)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Eventos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR EVENTO: Se crea un nuevo Evento a gestionar por la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Evento. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR EVENTO: Se muestra un listado con los Eventos disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos del Evento, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

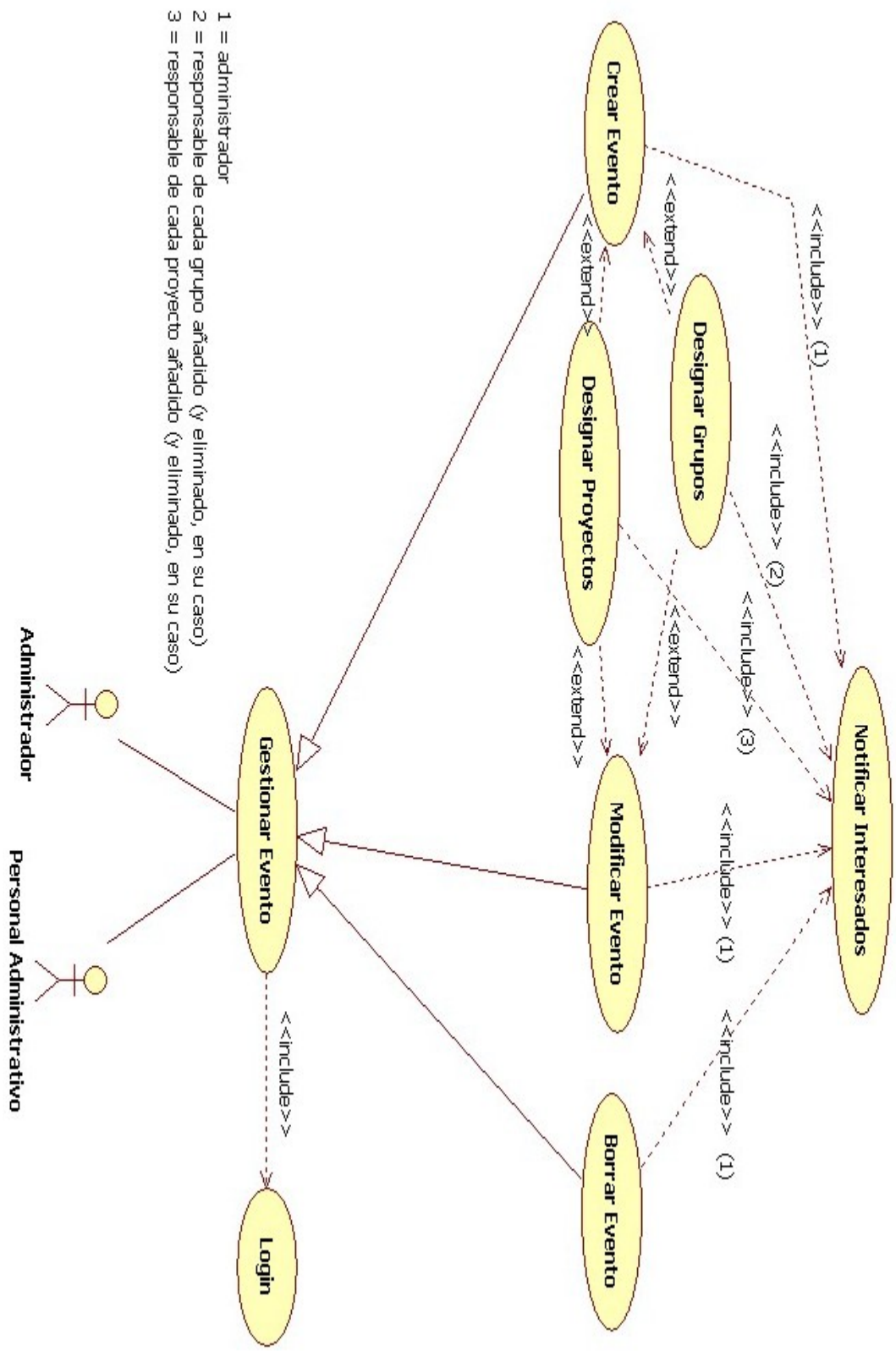
  - BORRAR EVENTO: Se elimina el Evento definitivamente de la aplicación. Se eliminarán también todas las relaciones de este evento, es decir, las relaciones con proyectos o grupos donde se anuncia serán eliminadas. No se guarda un histórico.

  - ASIGNAR PROYECTO: Tanto cuando se crea como cuando se modifica un evento, se permite la posibilidad de asignarlo a un proyecto de investigación. En este caso el evento será publicado para este proyecto.

  - ASIGNAR GRUPO: Cuando se crea o se modifica un evento, se permite la posibilidad de asignarlo a un grupo de investigación. En este caso el evento será publicado para ese grupo.

  - NOTIFICAR INTERESADOS: Cuando se realiza una operación sobre un Evento, se notifica automáticamente y mediante correo electrónico al ADMINISTRADOR de la aplicación.

    Diferencias con el modelo implementado: En el planteamiento inicial para eventos se determinó que cuando se crease/modificase un evento, este debía poseer una categoría asignada (el evento será mostrado en algún punto de la Web del departamento). En la implementación final se permite que los eventos no posean una categoría asignada. De esta forma se pueden cargar todos los eventos esperados en un período largo de tiempo y asignarles una categoría cuando esta esté definida.


1 = administrador
2 = responsable de cada grupo añadido (y eliminado, en su caso)
3 = responsable de cada proyecto añadido (y eliminado, en su caso)
Administrador
Personal Administrativo
Gestionar Evento
Crear Evento
Designar Grupos
Designar Proyectos
Modificar Evento
Borrar Evento
Notificar Interesados
Login
<<include>> (1)
<<include>> (2)
<<include>> (3)
<<extend>>

- **Gestión de Grupos: (Anexo B 10.2.1.4)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Grupos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR GRUPO: Se crea un nuevo Grupo para gestionarlo en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Grupo. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR GRUPO: Se muestra un listado con los Grupos disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Grupo, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR GRUPO: Se elimina el Grupo definitivamente de la aplicación. Se eliminarán también todas las relaciones de este grupo (eventos, líneas, etc.). Se eliminan también todas aquellas líneas de investigación que conforman el grupo y con esto, también proyectos que pertenecen a estas líneas si estos no están asignados a otras líneas de otros grupos no borrados. El sistema pide al usuario que modifique los grupos de todos aquellos empleados que pertenezcan al grupo a borrar. Si no se modifican estos empleados, no se borrará el Grupo. No se guarda un histórico.

  - CREAR LÍNEA: Cuando se crea (línea por defecto obligatoria) o modifica un Grupo, se permite al usuario de la aplicación crear nuevas líneas de investigación para el grupo con el que se opera. En todo caso habrá siempre una línea por defecto creada al crearse el grupo y que cuenta con sus mismos atributos. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear una nueva línea. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

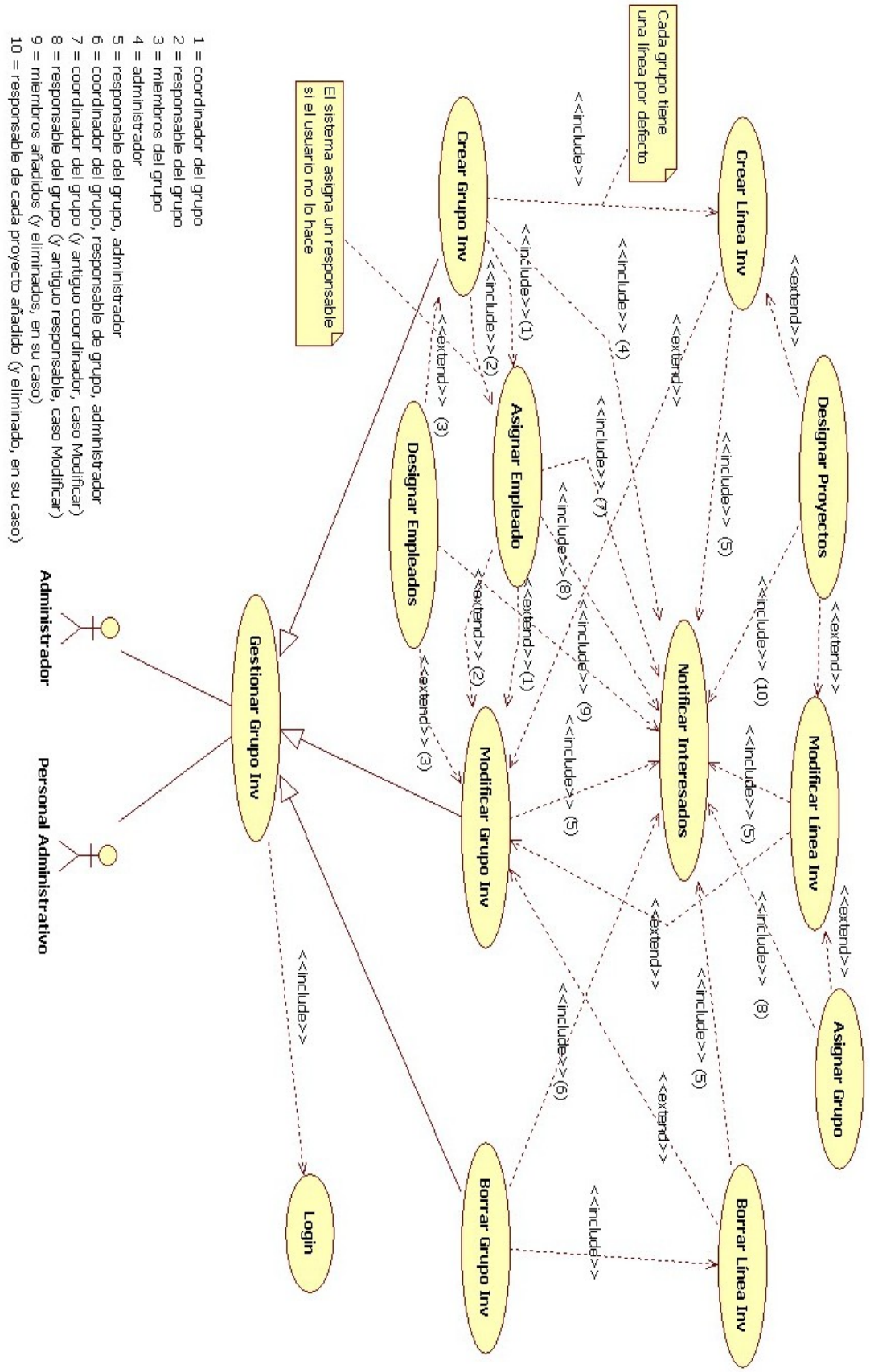
  - MODIFICAR LÍNEA: Se muestra un listado con las líneas disponibles en un grupo. Una vez seleccionada una de ellas, se mostrará un formulario con todos sus campos permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR LÍNEA: Se elimina la línea definitivamente de la aplicación. Dentro de un grupo podremos eliminar todas las líneas que este posea salvo la línea por defecto del grupo. Se eliminan también las relaciones que tenga la línea con otras entidades (Proyectos). No se guarda un histórico.

  - ASIGNAR GRUPO: Cuando se modifica una línea de investigación se le pide al usuario que asigne un grupo de investigación a esa línea.

- ASIGNAR EMPLEADO: Tanto cuando se crea como cuando se modifica un grupo de investigación, se da la posibilidad al usuario de asignar empleados que trabajan en el grupo.

- NOTIFICAR INTERESADOS: Se notificará mediante correo electrónico tanto al ADMINISTRADOR como al EMPLEADO responsable del Grupo cualquier operación sobre los grupos que controlan.


Cada grupo tiene una línea por defecto
El sistema asigna un responsable si el usuario no lo hace
1 = coordinador del grupo
2 = responsable del grupo
3 = miembros del grupo
4 = administrador
5 = responsable del grupo, administrador
6 = coordinador del grupo, responsable de grupo, administrador
7 = coordinador del grupo (y antiguo coordinador, caso Modificar)
8 = responsable del grupo (y antiguo responsable, caso Modificar)
9 = miembros añadidos (y eliminados, en su caso)
10 = responsable de cada proyecto añadido (y eliminado, en su caso)
Administrador
Personal Administrativo
Gestionar Grupo Inv
Crear Grupo Inv
Crear Línea Inv
Designar Proyectos
Asignar Empleado
Designar Empleados
Notificar Interesados
Modificar Grupo Inv
Modificar Línea Inv
Asignar Grupo
Borrar Grupo Inv
Borrar Línea Inv
Login
<<include>>
<<extend>>
<<include>> (1)
<<include>> (2)
<<include>> (4)
<<include>> (5)
<<include>> (6)
<<include>> (7)
<<include>> (8)
<<include>> (9)
<<include>> (10)
<<extend>> (1)
<<extend>> (2)
<<extend>> (3)

- **Gestión de la Home del Departamento: (Anexo B 10.2.1.9)**

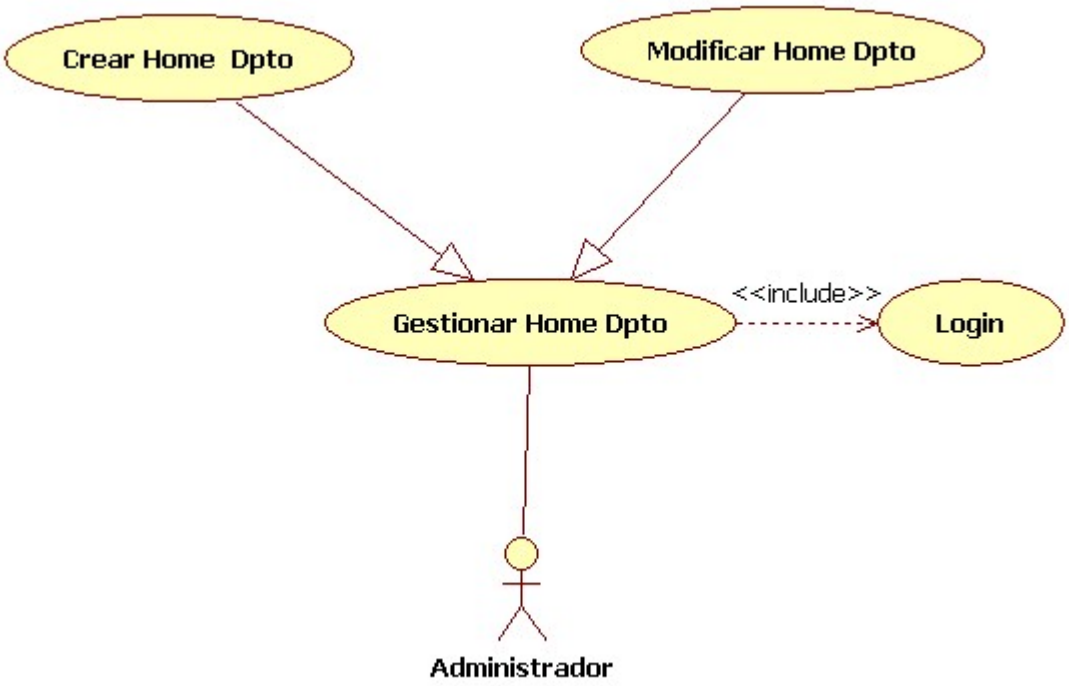
  Este sistema se limita al perfil de ADMINISTRADOR.

  - LOGIN: Para gestionar las Homes es necesario estar logado e identificado por el perfil mencionado anteriormente.

  - CREAR HOME: Se crea una nueva Home de departamento para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nueva Home. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR HOME: Se muestra un listado con las Homes existentes. Una vez seleccionada una de ellas, se mostrará un formulario con todos los campos de la Home, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR HOME: Se elimina la Home definitivamente de la aplicación. No se guarda un histórico.



- **Gestión de Tipos de Perfil: (Anexo B 10.2.1.11)**

  Este caso de uso se implementa sólo para esta versión y para experimentar con los cambios que esto supondría en la aplicación. Esto se explica en el anexo B sobre casos de uso.

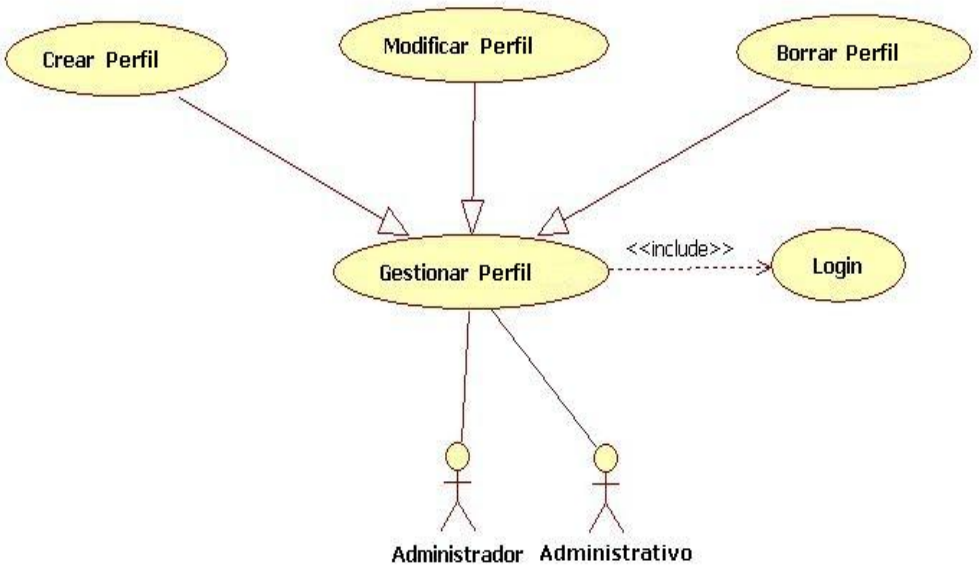
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Tipos de Perfil es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR PERFIL: Se crea un nuevo Perfil para ser gestionado por la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Perfil. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - MODIFICAR PERFIL: Se muestra un listado con los Perfiles disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Perfil, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Cualquier cambio realizado afectará a todos los empleados que posean este Perfil.

  - BORRAR PERFIL: Se elimina el Perfil definitivamente de la aplicación. Antes de poder eliminar un Perfil se pedirá al usuario que reasigne el perfil de todos los empleados que lo posean. En caso de no ser así no se eliminará el Perfil seleccionado. No se guarda un histórico.



- **Gestión de Proyectos 1: (Anexo B 10.2.1.3)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los Proyectos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR PROYECTO: Se crea un nuevo Proyecto a gestionar por la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Proyecto. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

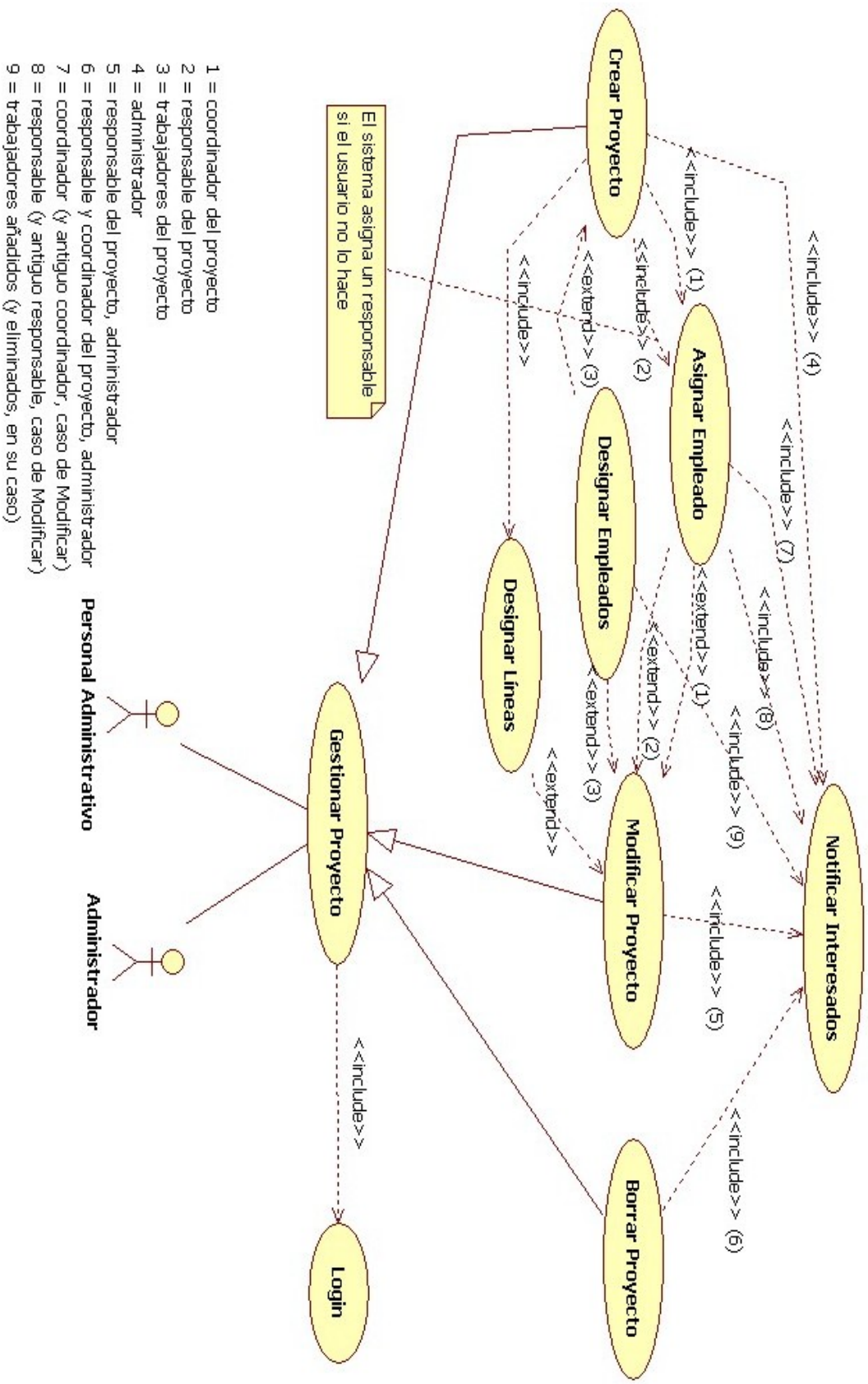
  - MODIFICAR PROYECTO: Se muestra un listado con los Proyectos disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos del Proyecto, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR PROYECTO: Se elimina el Proyecto definitivamente de la aplicación. Se eliminarán también todas sus relaciones con otras entidades de la aplicación (eventos, empleados, líneas). No se guarda un histórico.

  - ASIGNAR LÍNEA: Se permite al usuario asignar un proyecto a una línea de investigación. De esta forma un proyecto de investigación estará relacionado con el grupo al que pertenece la línea.

  - ASIGNAR EMPLEADO: Se permite al usuario asignar a empleados a proyectos, de esta forma representamos qué empleados trabajan en qué proyectos de investigación.

  - NOTIFICAR INTERESADOS: Se notificará mediante correo electrónico tanto al ADMINISTRADOR, ADMINISTRATIVO, EMPLEADO responsable del Proyecto y EMPLEADO coordinador del proyecto de cualquier operación sobre los proyectos que controlan.


Crear Proyecto
Asignar Empleado
Designar Empleados
Designar Líneas
Gestionar Proyecto
Modificar Proyecto
Notificar Interesados
Borrar Proyecto
Login
Personal Administrativo
Administrador
El sistema asigna un responsable si el usuario no lo hace
<<include>> (1)
<<include>> (2)
<<extend>> (3)
<<include>> (4)
<<include>> (5)
<<include>> (6)
<<include>> (7)
<<include>> (8)
<<include>> (9)
<<extend>> (1)
<<extend>> (2)
<<extend>> (3)
<<include>>
<<include>>
<<extend>>
<<include>>
1 = coordinador del proyecto
2 = responsable del proyecto
3 = trabajadores del proyecto
4 = administrador
5 = responsable del proyecto, administrador
6 = responsable y coordinador del proyecto, administrador
7 = coordinador (y antiguo coordinador, caso de Modificar)
8 = responsable (y antiguo responsable, caso de Modificar)
9 = trabajadores añadidos (y eliminados, en su caso)

- **Gestión de Proyectos 2: (Anexo B 10.2.1.3)**

  Este sistema se limita a los perfiles de EMPLEADO.

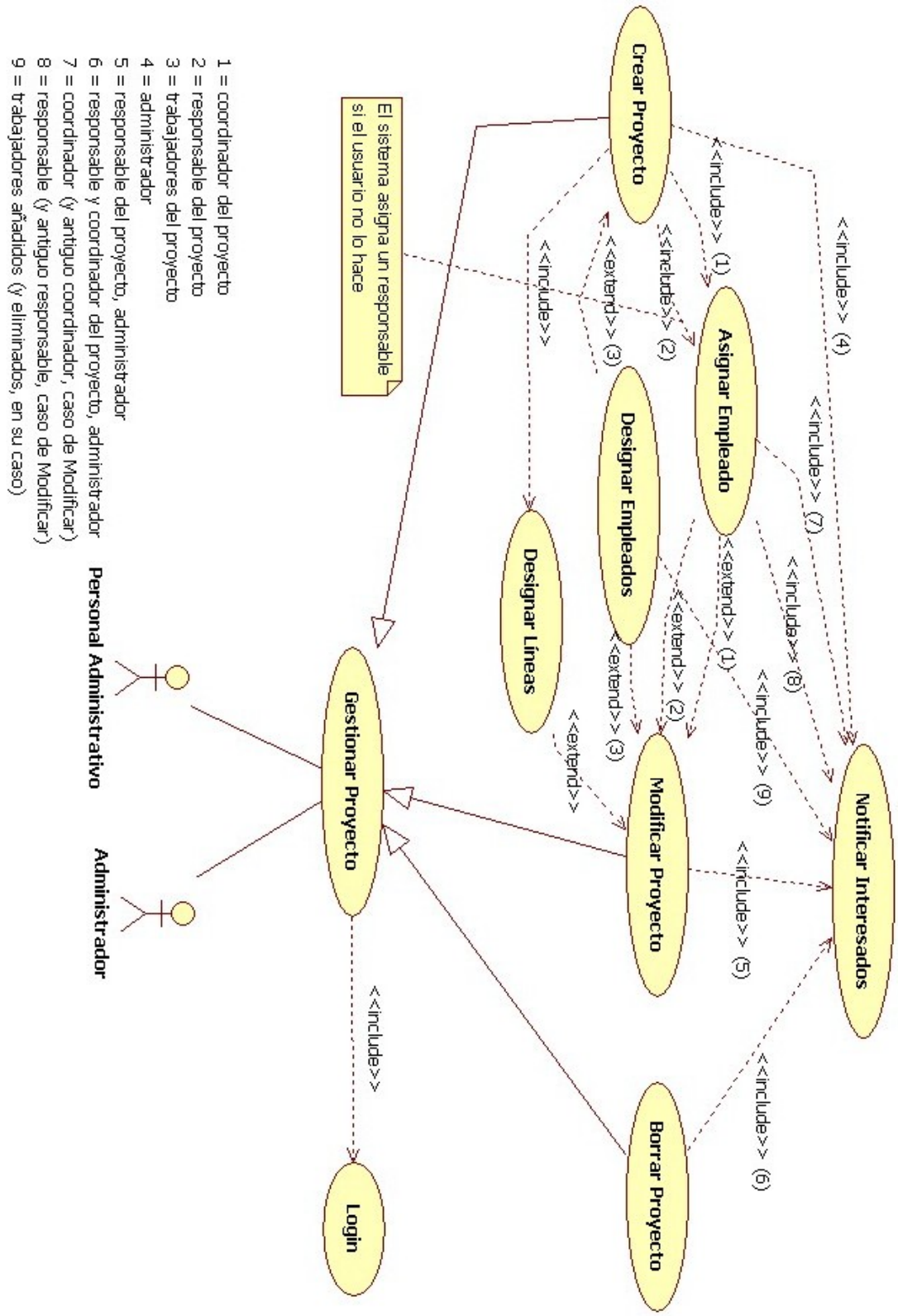
  - LOGIN: Para gestionar los Proyectos es necesario estar logado e identificado por el perfil mencionado anteriormente.

  - MODIFICAR PROYECTO: Se muestra un listado con todos los Proyectos que coordina el usuario. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos del Proyecto, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - ASIGNAR EMPLEADO: Se permite al usuario asignar a empleados a proyectos, de esta forma representamos qué empleados trabajan en qué proyectos de investigación. El usuario sólo tiene permitido modificar estos campos si es coordinador del proyecto.

  - ASIGNAR LÍNEA: Se permite al usuario asignar un proyecto a una línea de investigación. De esta forma un proyecto de investigación estará relacionado con el grupo al que pertenece la línea. El usuario sólo tiene permitido modificar estos campos si es coordinador del proyecto.

    Diferencias con el modelo implementado: En el planteamiento inicial se decidió que un empleado sólo podría ver los proyectos que coordinaba. En la implementación actual se permite poder visualizar todos los proyectos de la aplicación (con el objetivo de saber qué usuario era el director y responsable de todos los proyectos ya que esta información no estará disponible en la Web del departamento) pero sólo se le dará permisos de modificación sobre aquellos proyectos de los que el usuario es coordinador.


Crear Proyecto
Asignar Empleado
Designar Empleados
Designar Líneas
Gestionar Proyecto
Modificar Proyecto
Borrar Proyecto
Notificar Interesados
Login
Personal Administrativo
Administrador
El sistema asigna un responsable si el usuario no lo hace
<<include>> (1)
<<include>> (2)
<<extend>> (3)
<<include>> (4)
<<include>> (7)
<<include>> (8)
<<include>> (9)
<<extend>> (1)
<<extend>> (2)
<<extend>> (3)
<<include>> (5)
<<include>> (6)
<<include>>
<<extend>>
<<include>>
1 = coordinador del proyecto
2 = responsable del proyecto
3 = trabajadores del proyecto
4 = administrador
5 = responsable del proyecto, administrador
6 = responsable y coordinador del proyecto, administrador
7 = coordinador (y antiguo coordinador, caso de Modificar)
8 = responsable (y antiguo responsable, caso de Modificar)
9 = trabajadores añadidos (y eliminados, en su caso)

- **Gestión de Publicaciones: (Anexo B 10.2.1.5)**

  Este sistema se limita a los perfiles de ADMINISTRADOR, ADMINISTRATIVO y EMPLEADO.

  - LOGIN: Para gestionar las Publicaciones es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR PUBLICACIÓN: Se crea un conjunto indefinido de Publicaciones (depende del fichero de publicaciones subido por el usuario logado). Inmediatamente después de crear las publicaciones, se pedirá al usuario que establezca el autor/es de éstas.

  - MODIFICAR PUBLICACIÓN: Se muestra un listado con las Publicaciones disponibles. Una vez seleccionada una de ellas, se mostrará un formulario con todos sus campos y se permite que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

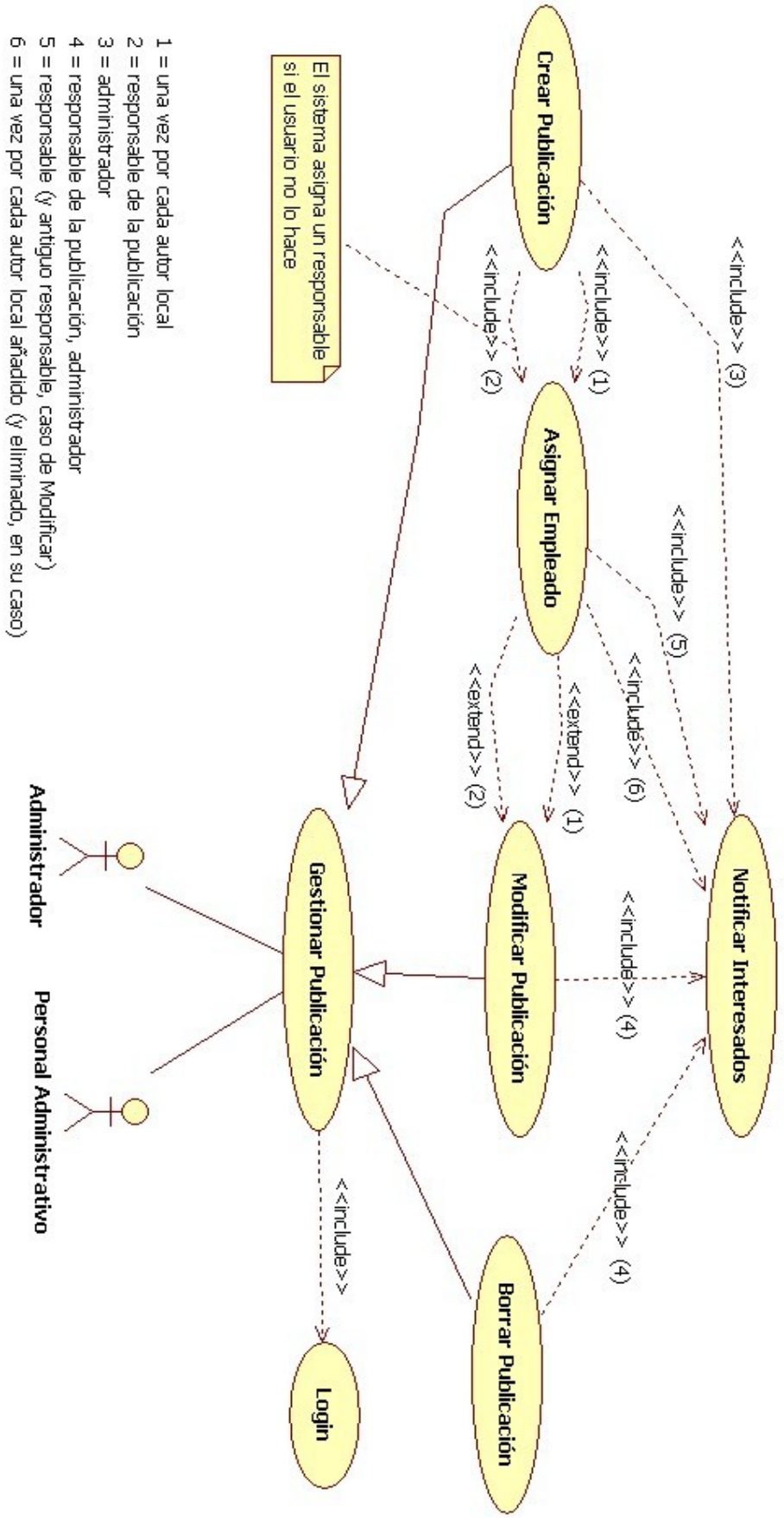
  - BORRAR PUBLICACIÓN: Se elimina la Publicación definitivamente de la aplicación. Se eliminarán también todas sus relaciones (empleados responsables, autores, etc). No se guarda un histórico.

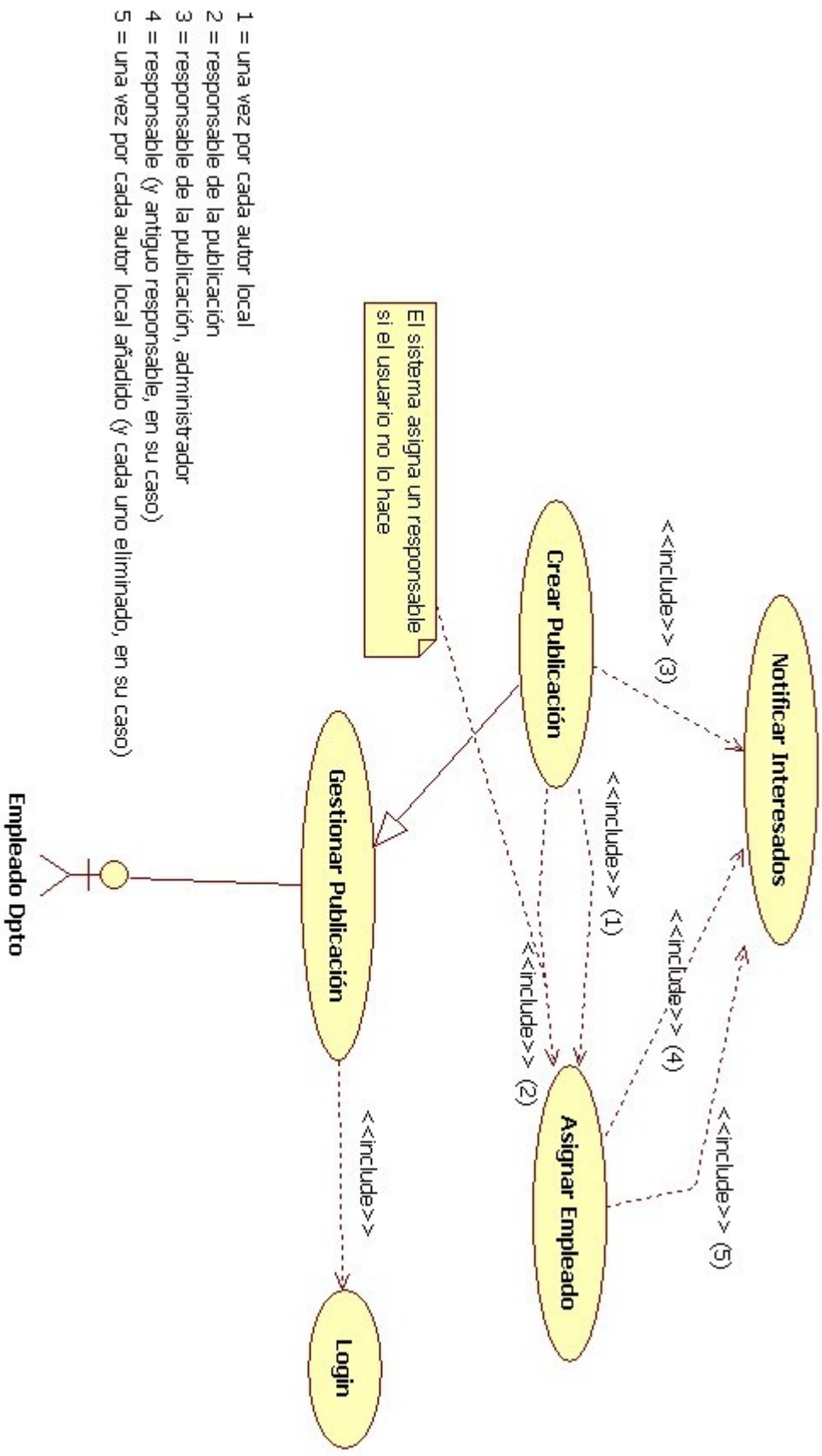
  - IMPORTAR DATOS: Se permite al usuario subir un fichero XML (para esta versión de la aplicación deberá ser BIBTEXML) que contenga todas las publicaciones que se quieran publicar en la WEB.

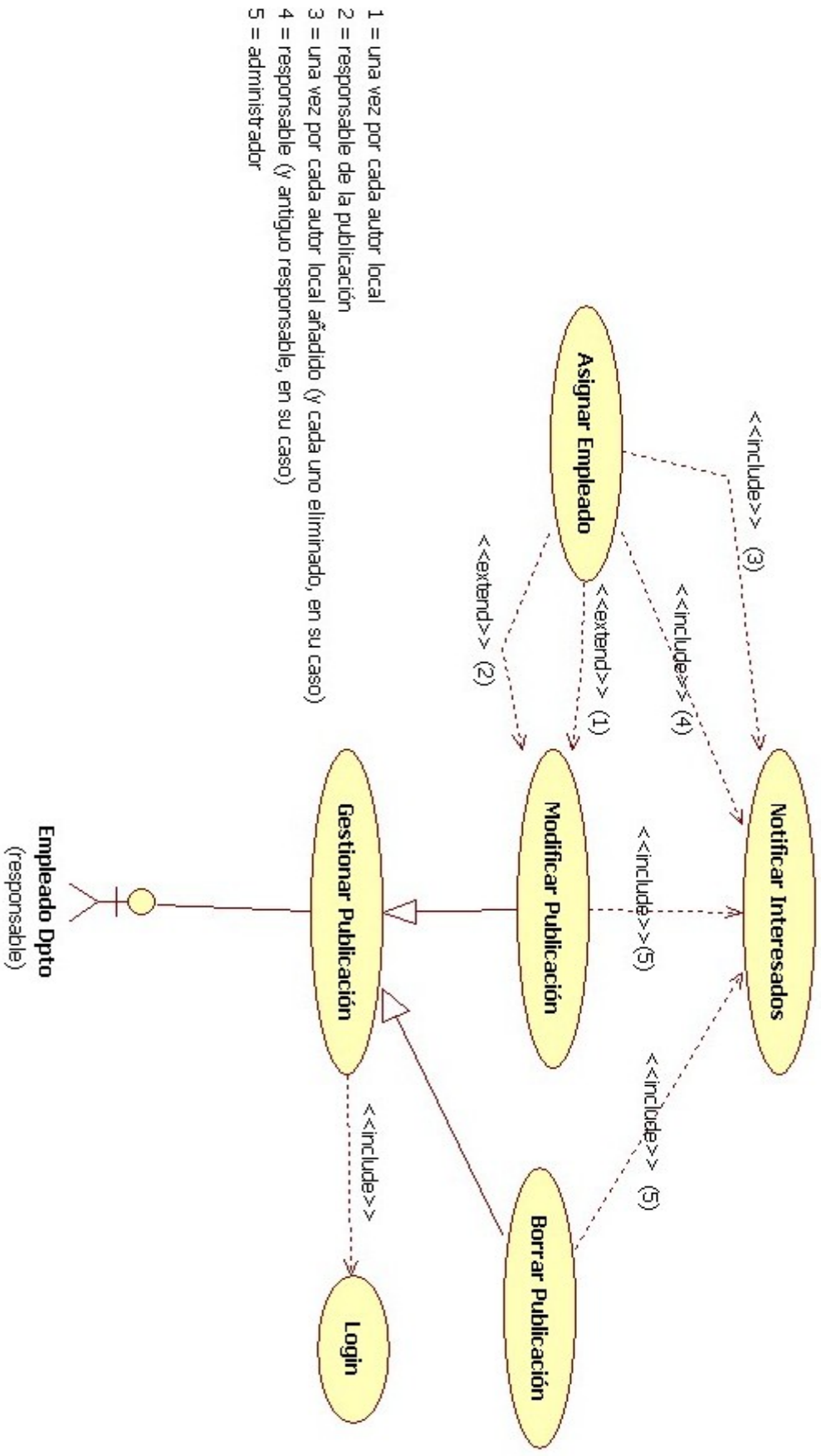
  - ASIGNAR EMPLEADO: Se permite al usuario establecer la persona responsable de una publicación (también los autores de ésta).

  - NOTIFICAR INTERESADOS: Se notificará mediante correo electrónico tanto al ADMINISTRADOR, EMPLEADO responsable de la Publicación y al EMPLEADO autores de la Publicación de cualquier operación sobre ella.

    Diferencias con el modelo implementado: El caso de uso de Gestión de Publicaciones es el que más modificaciones ha sufrido desde que se comenzó el diseño de la aplicación. Inicialmente se pensó en que sería el usuario quien mediante formularios html crearía una a una todas las publicaciones de las que fuera autor/coordinador. Este método se vió un poco tedioso para los usuarios de la aplicación por lo que se decidió que el usuario pudiese realizar una carga masiva de publicaciones. Además, se estableció el formato a seguir por estos ficheros de carga para esta primera versión, siendo BibTexml el elegido. Se ha tenido en cuenta también para iteraciones posteriores que el formato tenga diferentes fuentes.


Crear Publicación
Asignar Empleado
Notificar Interesados
Modificar Publicación
Gestionar Publicación
Borrar Publicación
Login
Administrador
Personal Administrativo
El sistema asigna un responsable si el usuario no lo hace
<<include>> (1)
<<include>> (2)
<<include>> (3)
<<include>> (5)
<<include>> (6)
<<extend>> (1)
<<extend>> (2)
<<include>> (4)
<<include>> (4)
<<include>>
1 = una vez por cada autor local
2 = responsable de la publicación
3 = administrador
4 = responsable de la publicación, administrador
5 = responsable (y antiguo responsable, caso de Modificar)
6 = una vez por cada autor local añadido (y eliminado, en su caso)


Notificar Interesados
Crear Publicación
Gestionar Publicación
Asignar Empleado
Login
Empleado Dpto
<<include>> (3)
<<include>> (1)
<<include>> (2)
<<include>> (4)
<<include>> (5)
<<include>>
El sistema asigna un responsable si el usuario no lo hace
1 = una vez por cada autor local
2 = responsable de la publicación
3 = responsable de la publicación, administrador
4 = responsable (y antiguo responsable, en su caso)
5 = una vez por cada autor local añadido (y cada uno eliminado, en su caso)


1 = una vez por cada autor local
2 = responsable de la publicación
3 = una vez por cada autor local añadido (y cada uno eliminado, en su caso)
4 = responsable (y antiguo responsable, en su caso)
5 = administrador
Asignar Empleado
Notificar Interesados
Modificar Publicación
Gestionar Publicación
Borrar Publicación
Login
Empleado Dpto
(responsable)
<<include>> (3)
<<include>> (4)
<<extend>> (2)
<<extend>> (1)
<<include>>(5)
<<include>> (5)
<<include>>

- **Gestión de Tesis 1: (Anexo B 10.2.1.6)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los datos de una Tesis es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

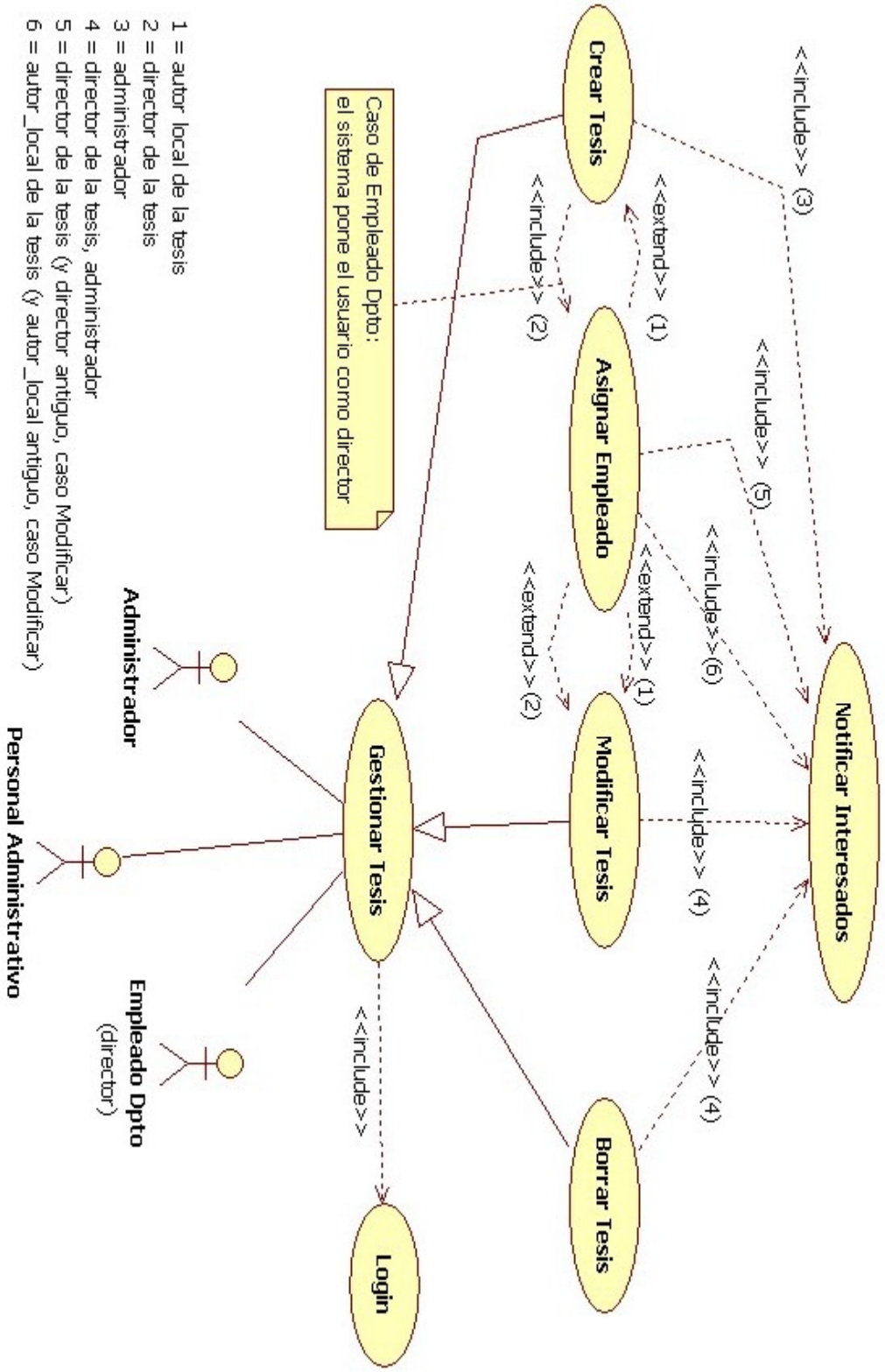
  - CREAR DATOS TESIS: Se crea una nueva Tesis para gestionar en la aplicación. Inicialmente se pregunta al usuario qué tipo de Tesis se pretende crear (Doctorado, PFC) y a partir de este punto se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear ese nuevo tipo de tesis. El seleccionar un tipo u otro conlleva la modificación del formulario a mostrar al usuario.

  - MODIFICAR DATOS TESIS: Se muestra un listado con las tesis disponibles. Una vez seleccionada una de ellas, se mostrará un formulario con todos los campos de la tesis, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR DATOS TESIS: Se elimina la Tesis definitivamente de la aplicación. No se guarda un histórico.

  - NOTIFICAR INTERESADOS: Se notificará mediante correo electrónico tanto al ADMINISTRADOR, EMPLEADO director de la tesis y al EMPLEADO autores de la tesis de cualquier operación sobre ella.

    Diferencias con el modelo implementado: Se ha modificado el modelo implementado para facilitar la interacción con el usuario. Ahora, se presenta al usuario una selección para crear PFCs o Doctorados. Según el empleado seleccione uno u otro, algunos de los campos serán autoseleccionados por la aplicación por ejemplo el campo "doctorado" de una tesis, es decir, será PFC cuando el campo doctorado sea falso y Doctorado en caso contrario. El resto de operaciones (modificar/borrar) también tendrán en cuenta si la Tesis se trata de un PFC o un Doctorado. Según el diseño se pretendía dejar el tipo de Tesis en manos del usuario pero este no tiene por qué saber que condiciones se tienen que dar para que la Tesis creada sea PFC o Doctorado por lo que finalmente se decidió implementarlo de esta nueva manera.


Crear Tesis
Asignar Empleado
Modificar Tesis
Borrar Tesis
Gestionar Tesis
Notificar Interesados
Login
<<include>> (3)
<<include>> (5)
<<include>> (6)
<<include>> (4)
<<include>> (4)
<<include>>
<<extend>> (1)
<<include>> (2)
<<extend>> (1)
<<extend>> (2)
Caso de Empleado Dpto:
el sistema pone el usuario como director
1 = autor local de la tesis
2 = director de la tesis
3 = administrador
4 = director de la tesis, administrador
5 = director de la tesis (y director antiguo, caso Modificar)
6 = autor_local de la tesis (y autor_local antiguo, caso Modificar)
Administrador
Personal Administrativo
Empleado Dpto
(director)

- **Gestión de Tesis 2: (Anexo B 10.2.1.6)**

  Este sistema se limita a los perfiles de EMPLEADO.

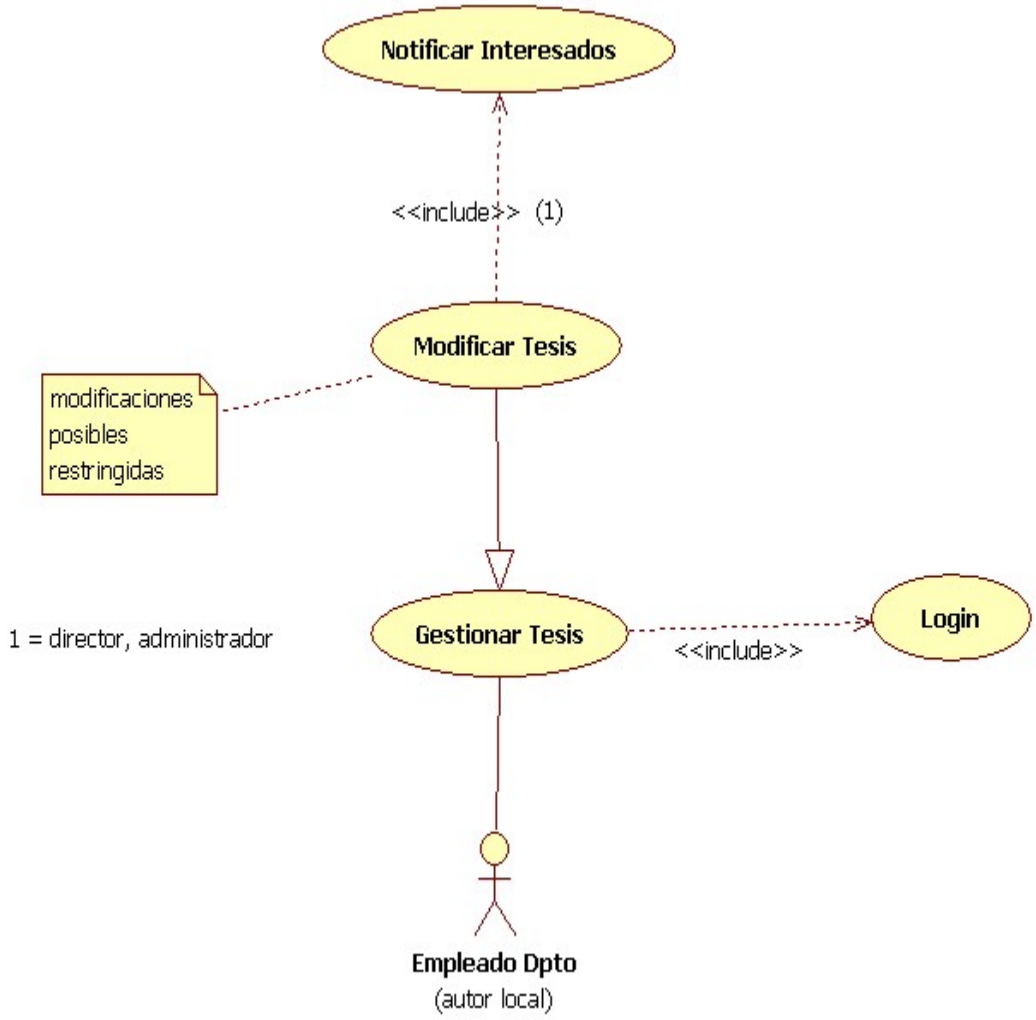
  - LOGIN: Para gestionar los datos de una tesis es necesario estar logado e identificado por el perfil mencionado anteriormente.

  - CREAR DATOS TESIS: Se crea una nueva Tesis para gestionar en la aplicación. Inicialmente se pregunta al usuario qué tipo de Tesis se pretende crear (Doctorado, PFC) y a partir de este punto se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear ese nuevo tipo de tesis. El seleccionar un tipo u otro conlleva la modificación del formulario a mostrar al usuario.

  - MODIFICAR DATOS TESIS: Se muestra un listado con las tesis disponibles para el usuario logado. Una vez seleccionada una de ellas, se mostrará un formulario con todos los campos de la tesis, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos.

  - BORRAR DATOS TESIS: Se elimina la Tesis definitivamente de la aplicación. No se guarda un histórico. Para poder borrar una tesis, el usuario EMPLEADO debe ser director de esta tesis.

  Diferencias con el modelo implementado: Ocurre igual que el caso de Gestión de Tesis para Administadores y Administrativos (En esta versión de la aplicación, se permite al autor_local que borre la tesis de la cual es autor. Esto debería ser trabajo sólo del director de la tesis, o de roles superiores de administración).


Notificar Interesados
<<include>> (1)
Modificar Tesis
modificaciones
posibles
restringidas
1 = director, administrador
Gestionar Tesis
<<include>>
Login
Empleado Dpto
(autor local)

- **Gestión de Tipos de Contrato: (Anexo B 10.2.1.11)**

  Este caso de uso se implementa sólo para esta versión y para experimentar con los cambios que esto supondría en la aplicación. Esto se explica en el anexo B sobre casos de uso.

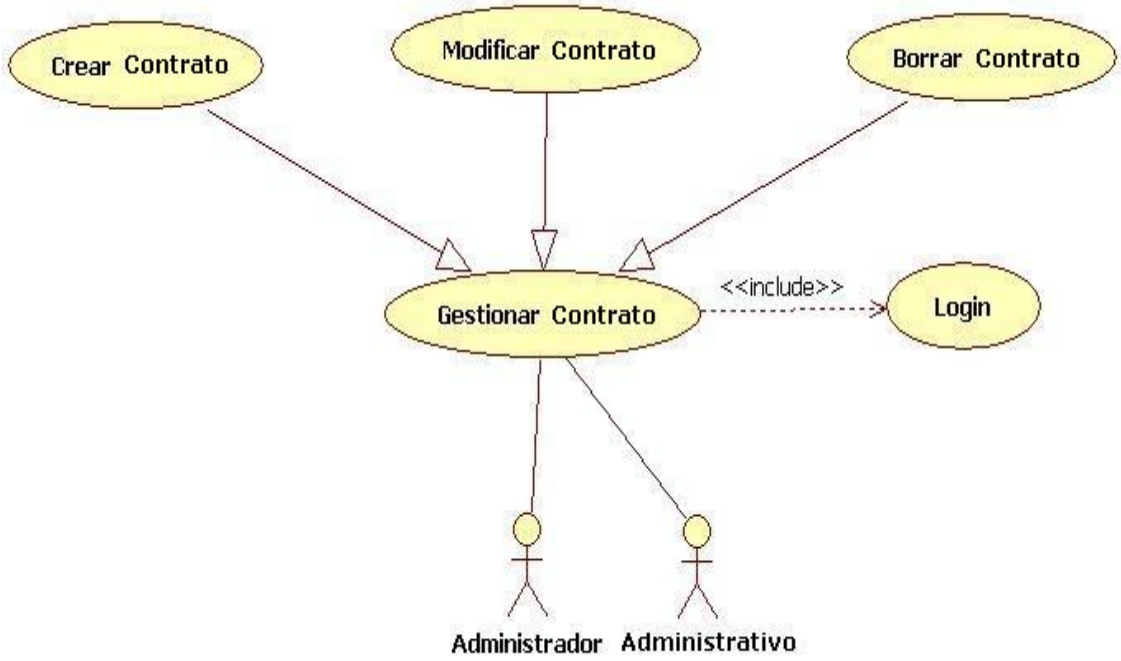
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los contratos es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR CONTRATO: Se crea un nuevo Tipo de Contrato para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Tipo de Contrato.

  - MODIFICAR CONTRATO: Se muestra un listado con los Tipos de Contrato disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Tipo de Contrato, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Cualquier cambio realizado afectará a todos los empleados que posean este contrato.

  - BORRAR CONTRATO: Se elimina el Tipo de Contrato definitivamente de la aplicación. Antes de poder eliminar un Tipo de Contrato se pedirá al usuario que reasigne el contrato de todos los empleados que poseen el Tipo de Contrato a eliminar. En caso de que no se modifiquen estos empleados, no se eliminará el Tipo de Contrato. No se guarda un histórico.



- **Gestión de Tipos de Proyecto: (Anexo B 10.2.1.11)**

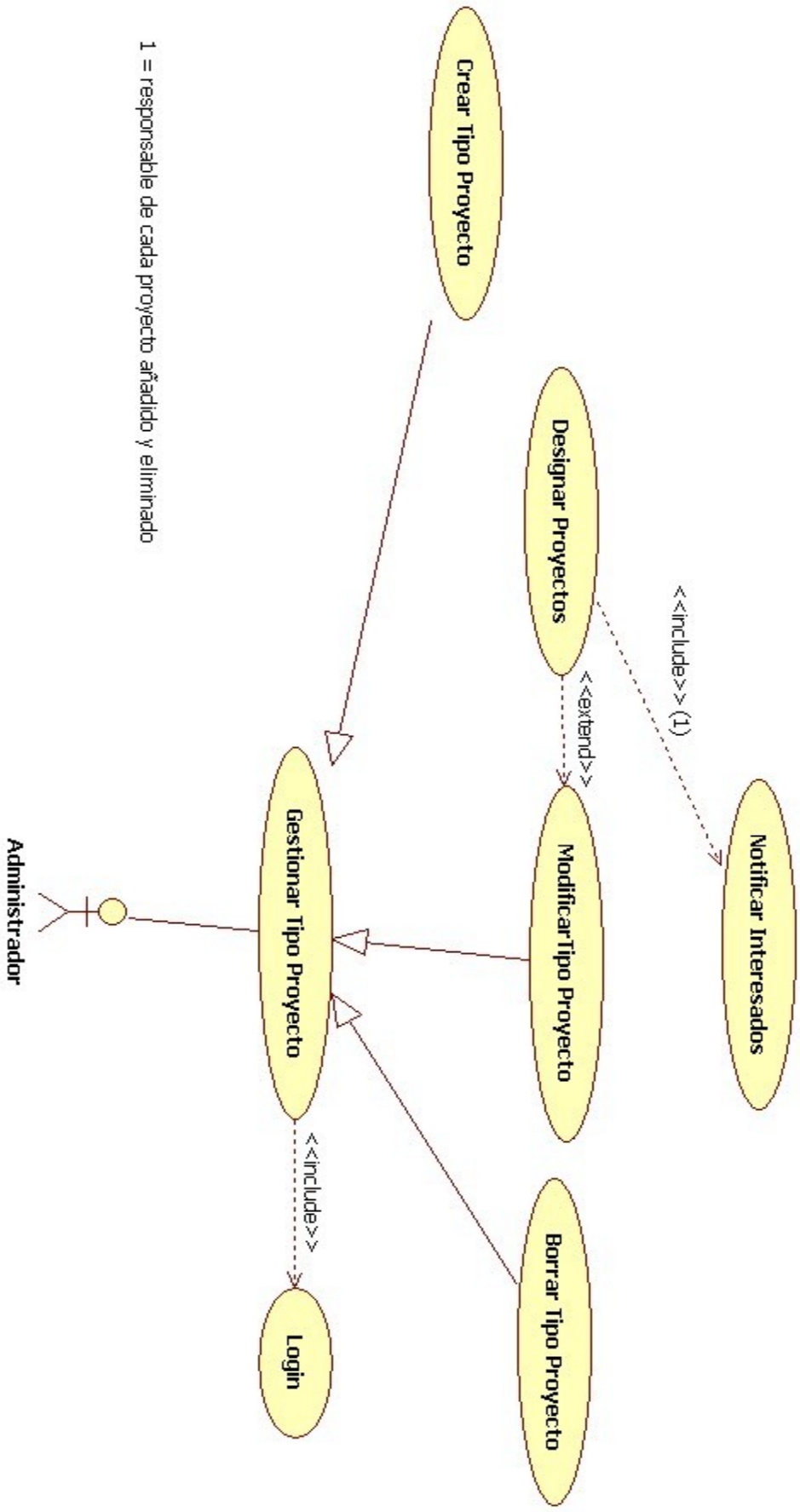
  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

  - LOGIN: Para gestionar los tipos de proyecto es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR TIPO DE PROYECTO: Se crea un nuevo Tipo de Proyecto para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear un nuevo Tipo de Proyecto.

  - MODIFICAR TIPO DE PROYECTO: Se muestra un listado con los Tipos de Proyecto disponibles. Una vez seleccionado uno de ellos, se mostrará un formulario con todos los campos de un Tipo de Proyecto, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Cualquier cambio realizado afectará a todos los proyectos de este tipo.

  - BORRAR TIPO DE PROYECTO: Se elimina el Tipo de Proyecto definitivamente de la aplicación. Antes de poder eliminar un Tipo de Proyecto se pedirá al usuario que modifique el tipo de proyecto de todos aquellos proyectos que posean el tipo de proyecto a eliminar. En caso de que el usuario no modifique estos tipos, no se eliminará el Tipo de Proyecto. No se guarda un histórico.


Crear Tipo Proyecto
Designar Proyectos
<<include>> (1)
Notificar Interesados
<<extend>>
Gestionar Tipo Proyecto
ModificarTipo Proyecto
Administrador
<<include>>
Borrar Tipo Proyecto
Login
1 = responsable de cada proyecto añadido y eliminado

- **Gestión de Web Personal: (Anexo B 10.2.1.12)**

  Este sistema se limita a los perfiles de EMPLEADO.

  - LOGIN: Para obtener la web personal, es necesario estar logado e identificado por el perfil mencionado anteriormente.

  - OBTENER WEB PERSONAL: El usuario accede a esta pantalla donde se le mostrará un fichero XML en el que está contenida su información personal. El usuario puede descargar este XML y modificarlo a su antojo para crear su página personal. De no ser así, también puede usar directamente este XML como página personal sin modificación alguna.

    Diferencias con el modelo implementado: Inicialmente se pensó que en este punto el usuario podía generar una página web html completa (bien generada por un proceso nocturno bien por la propia aplicación). En la implementación actual, y ya que para esta web personal se deja mayor libertad a los empleados del departamento, tan sólo se crea un fichero XML con la información personal del empleado que lo ha solicitado. De esta forma, con estos datos, los empleados pueden aplicar hojas de estilo o XSLT para crear una página personal a su gusto. En muchas ocasiones son los propios empleados los que crearán estas páginas para incluir datos que no vienen referenciados en la aplicación (p.e: cursos especiales, doctorados,etc.). Actualmente sólo se presentan los datos personales pero será sencillo modificar el formato del XML de salida para que incluya otros datos como proyectos, tesis, publicaciones, etc… en los que tiene participación el usuario.

- **Gestión de Titulaciones: (Anexo B 10.2.1.2)**

  Este sistema se limita a los perfiles de ADMINISTRADOR y ADMINISTRATIVO.

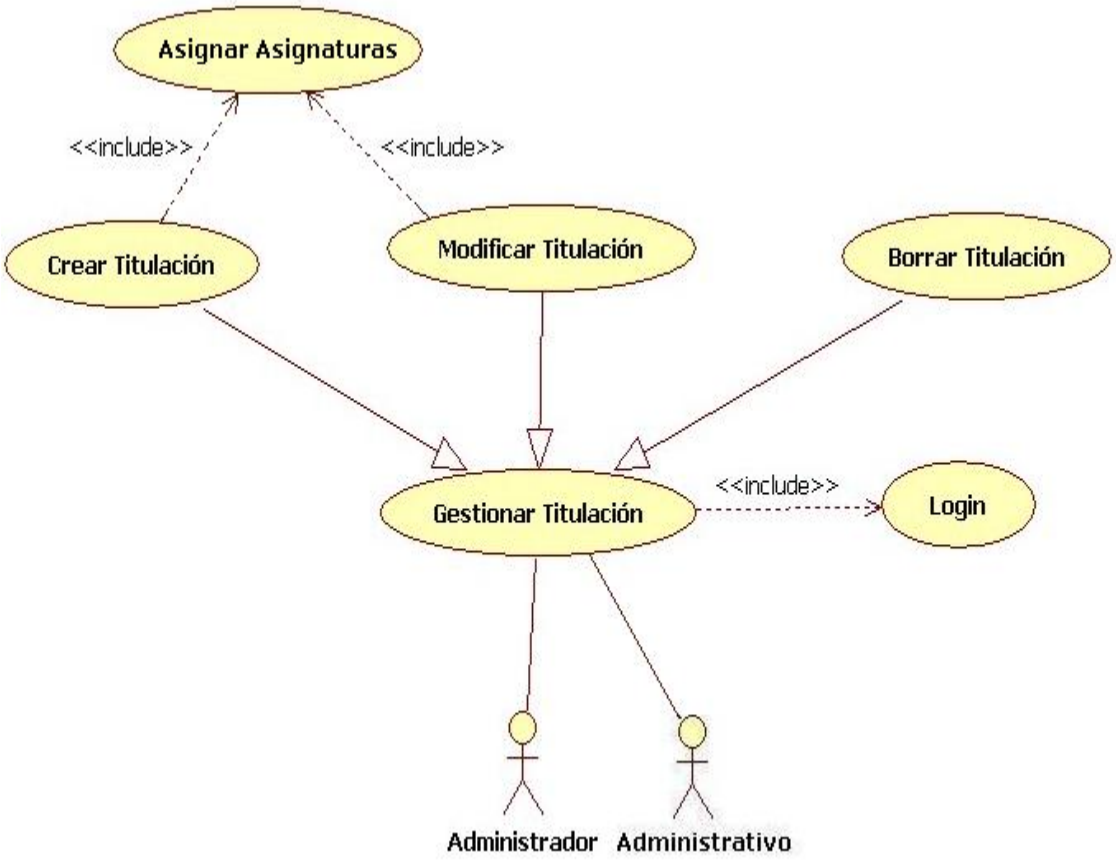
  - LOGIN: Para gestionar los tipos de proyecto es necesario estar logado e identificado por uno de los perfiles mencionados anteriormente.

  - CREAR TITULACIÓN: Se crea una nueva Titulación para gestionar en la aplicación. Se mostrará un formulario con todos los datos necesarios a ser introducidos por el usuario para crear una nueva Titulación. Se permite al usuario que cuando cree una Titulación indique qué asignaturas corresponden a esta Titulación.

  - MODIFICAR TITULACIÓN: Se muestra un listado con las Titulaciónes disponibles. Una vez seleccionada una de ellas, se mostrará un formulario con todos los campos de una Titulación, permitiendo que sean modificados por el usuario. En caso de que el usuario introduzca datos erróneos o que no sigan la política establecida, se notificará y se pedirá al usuario que reintroduzca estos datos. Se permite al usuario que introduzca las asignaturas que corresponden a la Titulación.

  - BORRAR TITULACIÓN: Se elimina la Titulación definitivamente de la aplicación. Se eleminarán también todas las relaciones que tenga la titulación con la aplicación. No se guarda un histórico.

    Diferencias con el modelo implementado: Inicialmente no se iba a implementar un caso "asignar asignaturas" ya que esto podía ser realizado desde cada asignatura. Pero a petición de los usuarios, se decidió incluir este nuevo caso de uso para centralizar el control de las asignaturas que pertenecen a una titulación en concreto.


Asignar Asignaturas
<<include>>
<<include>>
Crear Titulación
Modificar Titulación
Borrar Titulación
Gestionar Titulación
<<include>>
Login
Administrador
Administrativo

5. Diseño.

5.1 Estudio de la solución.

A continuación se detallarán todas las arquitecturas y tecnologías consideradas para el desarrollo de la aplicación:

La primera decisión tomada consiste en el modelo de la aplicación. Siguiendo el estándar más utilizado en la actualidad para el desarrollo de aplicaciones Web, se seguirá el Modelo-Vista-Controlador (MVC). Este estándar facilita la labor de los programadores al independizar los elementos de la aplicación por lo que los cambios realizados en ella, no se expanden a otras áreas de la aplicación.

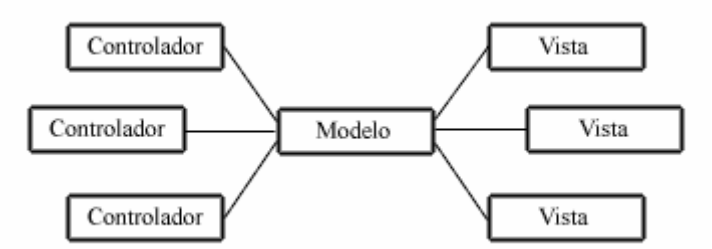
Definiciones de componentes:

El **Modelo** es el objeto que representa los datos del programa. Normalmente se trata de cada una de las entidades presentes en la b.b.d.d. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La **Vista** es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo. En java jsp, ruby rhtml, etc.

El **Controlador** es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

En la figura siguiente, vemos la arquitectura MVC en su forma más común. Hay un Modelo, varios Controladores que acceden a ese Modelo y hay varias Vistas de los datos del Modelo, que cambian cuando cambia el estado de ese Modelo.



Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado
- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

5.1.1 Arquitectura Distribuida.

En un principio se pensó los siguientes tipos de arquitectura para este proyecto:

- J2EE
- Ruby (Ruby On rails).
- PHP.
- Python.

Finalmente y por recomendación del director del proyecto se redujo la comparativa a sólo J2EE y Ruby.

5.1.1.1 J2EE (Java 2 Enterprise Edition).

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/Java_EE:

Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java con arquitectura distribuida, basándose en componentes de software modulares (O.O) ejecutándose sobre un servidor de aplicaciones. Este puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Algunas de las funcionalidades más importantes de J2EE son:

- Acceso a base de datos (JDBC)
- Utilizado por BEA, IBM, Oracle, Sun, y Apache Tomcat entre otros.
- Utilización de directorios distribuidos (JNDI).
- Acceso a métodos remotos (RMI/CORBA).

- Funciones de correo electrónico (JavaMail).
- Aplicaciones Web (JSP y Servlet)
- Uso de Beans.

La especificación original J2EE fue desarrollada por Sun Microsystems en 1997 comenzando con J2EE 1.3. El SDK de J2EE 1.3 fue liberado inicialmente como beta en Abril de 2001. La beta del SDK de J2EE 1.4 fue liberada por Sun en Diciembre de 2002.La especificación Java EE 5 fue liberada el 11 de Mayo de 2006.

Ventajas de J2EE

1.- Independiente del proveedor:

La plataforma J2EE ha sido creada con la participación de cientos de empresas de diversa índole y es, sin lugar a dudas una plataforma conjunta, no exclusiva de Sun o de ninguna otra compañía. Actualmente se han desarrollado una serie de herramientas (quizá las más conocidas sean JBuilder, de Borland, y ForteTM de la propia Sun) comerciales para implementar esta plataforma.

2.- Basado en Java:

El lenguaje en el que se basa J2EE es Java, un lenguaje orientado a objetos que alcanzó su madurez con la popularización de Internet y que es en cierta manera el heredero legítimo de C++. La expansión de este lenguaje entre la comunidad de programadores ha sido vertiginosa y se ha impuesto como el principal de los lenguajes de programación orientados a objetos. En el entorno académico e investigador, la enseñanza de Java ha reemplazado (y está reemplazando) a la enseñanza de lenguajes de programación estructurada como Pascal e incluso C que siempre se consideraban lenguajes de elección para la introducción a la programación.

3.-Multiplataforma y estable:

De forma resumida, Java es un lenguaje neutral, portable, robusto, estable, independiente de la plataforma, sencillo de aprender para programadores que hayan trabajado previamente con lenguajes orientados a objetos. Java puede utilizarse para realizar aplicaciones en múltiples plataformas hardware y sistemas operativos (Unix, Linux, OS/390, Windows, ó HP-UX entre otros sistemas operativos para ordenadores personales o estaciones de trabajo, y Palm OS ó EPOC entre otros sistemas operativos para dispositivos de telefonía móvil).

El modelo de desarrollo de J2EE

En la arquitectura J2EE se contemplan 4 capas en función del servicio y tipos de contenedores:

- **Capa de cliente**: también conocida como capa de presentación o de aplicación. Nos encontramos con componentes Java (applets o aplicaciones) y no-Java (HTML, JavaScript, etc.).
- **Capa Web**. Intermediario entre el cliente y otras capas. Sus componentes principales son los servlets y las JSP. Aunque componentes de capa cliente (applets o aplicaciones) pueden acceder directamente a la capa EJB, lo normal es que Los servlets/JSPs pueden llamar a los EJB.
- **Capa Enterprise JavaBeans**. Realmente se trataría de la capa de lógica de la aplicación. Nos centramos en las EJBs ya que son el modelo más extendido. Este modelo permite a múltiples aplicaciones tener acceso de forma concurrente a datos y lógica de negocio. Los EJB se

encuentran en un servidor EJB, que no es más que un servidor de objetos distribuidos. Un EJB puede conectarse a cualquier capa, aunque su misión esencial es conectarse con los sistemas de información empresarial. Sólo hace falta una capa EJB en aplicaciones de tamaño significativo. Si hicieramos la aplicación en Java EE no se utilizarían EJBs ya que con servlets y JSPs bastaría.

- **Capa de sistemas de información empresarial**. Esta capa engloba nuestras bases de datos, sistemas legacy, etc.

Al encontrar esta separación de capas en J2EE podemos concluir que es sencillo introducir cambios con esta arquitectura ya que estará separado por módulos y estos serán fáciles de cambiar o modificar por separado. Igualmente podemos decir que esta arquitectura permite desarrollar todas las partes por separado de forma simultánea lo que supone un gran ahorro en tiempo de desarrollo. Además, esto implica facilidad en cuanto a mantenibilidad, extensibilidad y reutilización de componentes.

En cuanto a la integración con otros sistemas, gran parte del modelo de J2EE (en especial los EJB) está basado en CORBA lo que quiere decir que es posible comunicarse sin ningún tipo de problema a través de IIOP con otras aplicaciones creadas en otros lenguajes diferentes de Java y viceversa. Aunque esto no es del todo cierto ya que más bien está basado en RMI para las comunicaciones distribuidos y que para las versiones posteriores, RMI se implementó sobre IIOP para tener compatibilidad con CORBA. En las últimas versiones también se pueden utilizar protocolos de Web Services

Por último otra tecnología que merece la pena nombrar es la de los conectores puente entre J2EE y sistemas legacy (por ejemplo un conjunto de ficheros de datos en COBOL). Los conectores tienen un API estándar que nos permite acceder a estos sistemas legacy de una manera transparente y sin apenas esfuerzo.

5.1.1.2 Ruby.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/Action_Mailer
- http://es.wikipedia.org/wiki/Ruby
- http://www.rubyonrails.org.es/

Ruby

Es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python, Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre

Desde su liberación pública en 1995, Ruby ha atraído devotos desarrolladores de todo el mundo. En el 2006, Ruby alcanzó reconocimiento masivo, formándose grupos de usuarios activos en las ciudades más importantes del mundo y llenando las capacidades de las conferencias relacionadas a Ruby.

El índice TIOBE, que mide el crecimiento de los lenguajes de programación, ubica a Ruby en la posición #13 del ranking mundial. Gran parte de su crecimiento se atribuye a la popularidad alcanzada por aplicaciones desarrolladas con Ruby, en particular el framework de desarrollo Web **Ruby on Rails.**

Ruby tiene un conjunto de otras funcionalidades entre las que se encuentran las siguientes:

- **Manejo de excepciones**: como Java y Python, para facilitar el manejo de errores.
- **Mark-and-sweep garbage collector:** No es necesario mantener contadores de referencias en bibliotecas externas.
- **Lenguaje Embebido:** Se permite que otros programas escritos en otros lenguajes de programación hagan llamadas a clases de Ruby así usarlo como lenguaje de scripting.
- **Extensibilidad:** Permite cargar bibliotecas de extensión dinámicamente si lo permite el sistema operativo.
- **Hilos:** Permite muti-threading independientemente del sistema operativo.
- **Portable**: se desarrolla mayoritariamente en GNU/Linux, pero corre en varios tipos de UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.

Ruby On Rails

Ruby on Rails fue escrito por David Heinemeier Hansson a partir de su trabajo en Basecamp, una herramienta de gestión de proyectos, por 37signals. Fue liberado al público por primera vez en Julio de 2004.

- Ruby on Rails 1.0 fue publicado el 13 de diciembre de 2005.
- Ruby on Rails 1.1 fue publicado el 28 de marzo de 2006.
- Ruby on Rails 1.2 fue publicado el 18 de enero de 2007.
- Ruby on Rails 2.0 fue publicado el 7 de diciembre de 2007.
- Ruby on Rails 2.1 fue publicado el 1 de junio de 2008.

Ruby on Rails, también conocido como RoR o Rails es un framework de aplicaciones Web de código abierto escrito en el lenguaje de programación Ruby, que sigue la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración.

El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Esta metaprogramación se define como "programas que escriben otros programas". Ruby, por ejemplo, permite que las clases incluyan otras clases por lo que los métodos de las clases incluidas estarán disponibles sin llamar a estas clases y si la necesidad de tener que escribir el código del método en la clase llamada. Otro ejemplo de metaprogramación en Ruby es definir un objeto "me" como "self" de esta forma si llamamos "me.metodo" estaremos haciendo lo mismo que "self.metodo" de ahí que se diga que la lectura puede ser mucho más legible al poder definir palabras más similares al lenguaje humano. Gracias a la metaprogramación de Ruby, sentencias como "belongs_to" o "has_many" estan disponibles para todas las entidades que se creen en la aplicación, de esta forma solo hay que indicar cual es la entidad/es a las que queremos que se aplique este metodo y ruby se encargará de "escribir" el código que se necesita para implementar esas funcionalidades.

Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de librerías y aplicaciones Ruby.

Ruby On rails: Gems y CookBooks

Las librerias de ruby on rails son conocidas como "Gems" (gemas), y es el equivalente a trabajar con los "jars" de Java. Como se ha dicho antes, Rails se distribuye a través de RubyGems. Este es el punto desde el cual podemos actualizar o encontrar todas las gemas que se necesiten.

Existen patrones de diseño ya creados en Ruby on Rails. Estos patrones son conocidos como "recipes" y en ellos están detallados la mayoría de casos que se pueden presentar a la hora de crear una aplicación Web. Es el caso de envío de mails, login, parseo de XML, etc. estas recetas se guardan en los "cookbooks" de ruby on rails y están disponibles de forma gratuita, en su mayoría, para todos los programadores.

La filosofía de **Ruby on Rails** se puede resumir en tres principios fundamentales:

· **DRY**: Don't Repeat Yourself
· **COC**: Convention Over Configuration
· **Agilidad**

DRY: Don't Repeat Yourself traducido al castellano como "No te repitas" significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Por ejemplo, en ActiveRecord, las definiciones de las clases no necesitan especificar los nombres de las columnas; Ruby puede averiguarlos a partir de la propia base de datos, de forma que definirlos en el código sería redundante.

COC: Convention Over Configuration que se traduce como "Convención sobre configuración" significa que el programador sólo necesita definir aquella configuración que no es convencional. Por ejemplo, si hay una clase Empleado en el modelo, la tabla correspondiente de la base de datos es Empleados, pero si la tabla no sigue la convención (por ejemplo Tesis) debe ser especificada manualmente (set_table_name "Tesis"). Así, cuando se diseña una aplicación partiendo de cero sin una base de datos preexistente, el seguir las convenciones de Rails significa usar menos código (aunque el comportamiento puede ser configurado si el sistema debe ser compatible con un sistema heredado anterior).

Agilidad: este punto de la filosofia es heredado de los otros dos puntos. Si sabemos utilizar correctamente el DRY y el COC agilizaremos de forma notable el desarrollo de nuestros aplicativos, consiguiendo reducir al máximo nuestro tiempo de desarrollo.

Ruby On rails: ORM (Object Role Modeling) -> Active Record

ORM: es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

En Ruby On Rails, Active Record conecta los objetos de negocio y las tablas de b.b.d.d. para crear un modelo persistente donde la lógica y la información son presentadas en un mismo paquete. Es una implementación de ORM.

Un objeto que representa a una fila dentro de una tabla o vista de la b.b.d.d., encapsula el acceso a b.b.d.d. y añade la lógica de ese tipo de objeto sobre la información que este contiene.

La principal contribución de Active Record al modelo original es el de eliminar de éste dos problemas: la falta de asociaciones y la herencia añadiendo solamente un conjunto de macros. Además de ser un paquete "stand-alone" de Ruby, Active Record también es el MODELO del framework para aplicaciones web Rails.

Características:

- **No se necesita Metadata**
  - Desaparecen los ficehros XML de configuración. Active Record se configura al vuelo sin necesidad de una fase de construcción (build).
- **Soporte de Bases de Datos**
  MySQL, PostgreSQL (7.4+), and SQLite son soportados directamente. Para el resto de b.b.d.d es necesario escribir un nuevo Adaptador de Bases de Datos.
- **Multihilo**
  Active Record permite el usao de servidores web escritos en Ruby que pueden manejar peticiones usando hilos como son, por ejemplo, WEBrick y Cerise.
- **Permite Transacciones**
  Active Record utiliza transacciones para asegurar que los borrados dependientes se llevan a acabo automáticamente. También permite escribir métodos incluidos dentro de una transacciónl.
- **Asociaciones Sencillas**
  Las asociaciones entre clases se hacen de una forma sencilla usando macros escritas en un lenguaje natural como por ejemplo: "has_many" y "belongs_to".

5.1.1.3 PHP.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/.php
- http://www.php.net/

PHP es un lenguaje de programación intrepretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTL+.

PHP es un acrónimo recursivo que significa ***P****HP* ***H****ypertext* ***P****re-processor* (inicialmente PHP Tools, o, *P*ersonal *H*ome *P*age Tools). Fue creado originalmente por Rasmus Lerdof en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, aunque el número de sitios en PHP ha empezado a compartir su cuota con otros lenguajes desde hace unos años. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web. La más reciente versión principal del PHP fue la versión 5.2.6 de 1 de mayo de 2008.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructuda, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la línea de órdenes, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (*Command Line Interface*).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C# VB.NET como lenguajes), a ColdFusion de la compañía Adobe (antes Macromedia), a JSP/Java de Sun Microsystems, y a CGI/Perl. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (entorno de desarrollo integrado) comercial llamado Zend Studio. Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno integrado de desarrollo para PHP, denominado **Delphi for PHP**. Existe un módulo para Eclipse uno de los IDE más populares.

Historia:

PHP fue originalmente diseñado en Perl, en base a la escritura de un grupo de CGI binarios escritos en el lenguaje C por Rasmus Lerdorf en el año 1994 para mostrar su C.V. y guardar ciertos datos, como la cantidad de tráfico que su página web recibía. El 8 de junio de 1995 fue publicado "**P**ersonal **H**ome **P**age Tools".

Características:

- **Favorables:**

1. Es un lenguaje multiplataforma.
2. Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
3. Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
4. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
5. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
6. Permite las técnicas de Progrmación Orientada a Objetos.
7. Biblioteca nativa de funciones sumamente amplia e incluida.
8. No requiere definición de tipos de variables.
9. Tiene manejo de excepciones (desde php5).

- **Desfavorables:**

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aún estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño, MODELO VISTA CONTROLADOR (o MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

5.1.1.4 Python.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/Python
- http://www.python.org/

Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1990. Se compara habitualmente con TCL, Perl, Scheme, Java y Ruby. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. La última versión estable del lenguaje es la 2.6 (01 de octubre de 2008) (Se anunció la llegada de la versión 3.0 para agosto de 2008, aunque fue el 17 de septiembre de dicho año cuando se lanzó la primera versión rc1 -release candidate- de dicha versión).

Python es considerado como la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar.

Python permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, Qt entre otros...

Python es un lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

Características y Paradigmas:

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Otros muchos paradigmas más están soportados mediante el uso de extensiones. Python usa tipo de dato dinámico y reference counting para el manejo de memoria. Una característica importante de Python es la resolución dinámica de nombres, lo que liga nombres de métodos y de variables en tiempo de ejecución del programa (también llamado ligadura dinámica de métodos).

Otro objetivo del diseño del lenguaje era la facilidad de extensión. Nuevos módulos se pueden escribir fácilmente en C o C++. Python puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan de una interfaz programable. Aunque el diseño de Python es de alguna manera hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia del Lisp como puede ser Scheme.

5.1.1.5 Decisión.

Finalmente, tras evaluar las opciones de arquitectura de las que disponemos. La aplicación de control de datos se ha decidido implementarla mediante Ruby On Rail. La vertiente de presentación de datos en cambio realizará los accesos a base de datos mediante J2EE, la generación de xml mediante XMLBEANS y la presentación final mediante XSLT.

A pesar del coste de aprendizaje de Ruby y el uso del framework Rails, finalmente se decidió que esta parte de la aplicación fuese escrita en Ruby on Rails.

¿Por qué se ha elegido Ruby On Rails?

- **Coste de desarrollo:** Podemos encontrar numerosos entornos de desarrollo o IDEs para Ruby on Rails que además son gratuitos. En este caso se utilizará la suite de Aptana, que para los que hayan programado en Eclipse, será muy familiar. Además, debido a la naturaleza de RoR (script para generar el esqueleto de la aplicación, scripts para generar la base de datos, scripts para generar controllers, models, etc.), se pueden desarrollar aplicaciones web en un tiempo relativamente corto. Como contraposición a este punto, hay que decir que uno de los principales inconvenientes de usar RoR es el desplieque de la aplicación final. Esto sucede debido a problemas de estabilidad con los servidores HTTP (WEBrick, Mongrel), rendimiento, conexión a la base de datos defectuosa, etc.

- **Portabilidad:** Esta arquitectura es funcional en los Sistemas Operativos actuales más conocidos.

- **Mantenibilidad:** Se eligión RoR ya que al implementar MVC su mantenimiento se hace sencillo.

- **Decisión de Proyecto:** Para establecer una línea divisoria clara entre ambos proyectos, presentación de datos y control de estos, el director del proyecto planteó una diferenciación mediante la arquitectura a utilizar. Esta decisión es suficiente para que Ruby on Rails se impusiese como arquitectura de esta parte de la aplicación y como arquitectura del proyecto de control de datos.

- **Experimentación con nuevas tecnologías:** Otra razón importante es que se deseaba experimentar con RoR para clarificar sus ventajas e inconveniente con respecto a soluciones parecidas sobre un caso de estudio bien conocido.

5.1.2 Servidor de Aplicaciones.

5.1.2.1 Jboss.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/JBoss
- http://www.jboss.com
- http://www.jboss.org

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en Abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios. JBoss implementa todo el paquete de servicios de J2EE (EJB, JMS, JTS/JTA, Servlets/JSP, JNDI, etc.) y también ofrece características tales como los clustering, JMX, Web Services y la integración IIOP, y la principal característica es que JBoss tiene licencia LGPL, puede libremente usarse sin costo alguno (la versión básica sin soporte continuo) en cualquier aplicación comercial o ser redistribuido.

Servidor de aplicaciones de Jboss (Jboss AS)

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones java, aplicaciones Web, Portales y e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado, y distribuido sin restricciones por la licencia. Por este motivo es una de las plataformas más populares de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas. Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.
- Ayuda profesional 24x7 de la fuente.
- Soporte completo para JMX.
- Soporte para Hibernate.
- Soporte para EJB (versión 2 y 3).

5.1.2.2 BEA Weblogic.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/BEA_WebLogic
- http://www.oracle.com/bea/index.html

BEA WebLogic es un servidor de aplicaciones J2EE y también un servidor web HTTP de BEA Systems de San José, California, para Unix, Linux, Microsoft Windows, y otras plataformas.

WebLogic puede utilizar Oracle, DB2, Microsoft SQL Server, y otras bases de datos que se ajusten al estándar JDBC. El servidor WebLogic es compatible con WS-Security y cumple con los estándares de J2EE 1.3 desde su versión 7 y con la J2EE 1.4 desde su versión 9.

BEA WebLogic Server es parte de BEA WebLogic Platform. Los demás componentes de esta plataforma son:

a) Portal, que incluye el servidor de comercio y el servidor de personalización (construido sobre un motor de reglas producido también por Bea, Rete),
b) Weblogic Integration,
c) Weblogic Workshop, una IDE para Java, y
d) JRockit, una máquina virtual Java (JVM) para CPUs de Intel

WebLogic Server incluye interoperabilidad .NET y admite las siguientes capacidades de integración nativa:

- Mensajería nativa JMS a escala de empresa
- J2EE Connector Architecture
- Conector WebLogic/Tuxedo
- Conectividad COM+
- Conectividad CORBA
- Conectividad IBM WebSphere MQ

BEA WebLogic Server Process Edition también incluye Business Process Management y funcionalidad de mapeo de datos.

WebLogic admite políticas de seguridad administradas por Security Administrators. El modelo de seguridad de BEA WebLogic Server incluye:

- Separar la lógica de aplicaciones de negocio del código de seguridad
- El rango completo de cobertura de seguridad tanto para los componentes J2EE y no J2EE.

5.1.2.3 Servidor Mongrel.

Las siguientes definiciones han sido tomadas de:

- http://en.wikipedia.org/wiki/Mongrel_(web_server)
- http://mongrel.rubyforge.org/

Mongrel es una librería http open-source HTTP y un servidor Web para las aplicaciones Web de Ruby escrito por Zed A. Shaw. Una característica significativa de Mongrel es que usa html plano, como FastCGI o SCGI, para comunicarse con otros servidores que han podido ser desplegados por delante de él.

La gran mayoría piensa que Mongrel es más rápido y más estable que WEBrick, y más fácil de configurar que Apache. La configuración más común cuando se utiliza Mongrel es lanzar Apache 2.2 como balanceador de carga utilizando "mod_proxy_balancer" conjuntamente con varias intancias de Mongrel, con cada instancia corriendo en puertos diferentes. Esto es algo que puede ser configurado muy fácilmente utilizando la herramienta "mongrel_cluster management utility". Apache puede redirigir las request entrants entre los procesos de Mongrel disponibles y, con una configuración cuidada, puede incluso servir contenido estático sin la necesidad de delegarselo a los servidores Mongrel.

Para aquellos que quieran evitar un uso conjunto con Apache,es posible desplegar un cluster de Mongrel con un servidor Web alternativor, como nginx o lighttpd, y un balanceador de carga como por ejemplo Pound o una solución basada en hardware como la F5 Networks BIG-IP.

Capacidad de Stand-alone

Uno de los principales problemas de Mongrel es que no soporta threads y por ello a veces se da la necesidad la necesidad de actuar por detrás de otros servidores. Lo normal es establecer un servidor lighttpd que manejan los hilos y después redirige las peticiones a varios Mongrel que tiene funcionando por debajo. Esta configuración plantea un problema de mayor consumo de recursos, tiempo de configuración, etc.

Mongrel es capaz de servir sitios de Ruby on Rails sin la necesidad de ningún otro servidor web, aunque al ser una aplicación mono-hilo esta configuración es incompatible con todas las cargas excepto por las ligeras.

5.1.2.4 Servidor Lighttpd.

Las siguientes definiciones han sido inspiradas de:

- http://www.lighttpd.net/
- http://es.wikipedia.org/wiki/Lighttpd

Lighttpd es un servidor Web que ha sido diseñado para ser seguro, rápido, compilable y flexible mientras se optimiza para entornos críticos en cuanto a la velocidad.

Tiene una huella de memoria (footprint) muy pequeña comparada con la de otros servidores, carga ligera de CPU y sus niveles de velocidad máximos hacen que lighttpd se ajuste a servidores que sufren problemas de carga o para servir contenido estático a parte del contenido dinámico. Lighttpd es un software gratuito y open source y está distribuido bajo las licencia BSD. Lighttpd funciona sobre Linux, otros S.S.O.O. como UNIX y Microsoft Windows (bajo Cygwin). Bajo Windows, puede ser controlado usando el programa Lighty Tray.

- Balanceador de carga FastCGI, SCGI y soporte de HTTP-proxy.
- Soporte de chroot.
- Servidor Web basado en select ()-/poll ()-based.
- Soporte de esquemas de notificación de eventos más como kqueve y epoll.
- Reescritura condicional (mod_rewrite).
- Soporte de SSL y TLS, via OpenSSL.
- Autenticación contra servidores LDAP.
- Estadísticas rrdtool.
- Descarga basada en reglas con posibilidad de un script que maneje la autenticación.
- Soporte en el lado del servidor.
- Hosting virtual flexible.
- Soporte de módulos.
- Lenguaje de Meta Caché (actualmente está siendo reemplazado por mod_magnet).
- Soporte mínimo de WebDAV.
- Soporte de Servlet (AJP, en versión 1.5.x en adelante).
- Compresión HTTP usando mod_compress y el nuevo mod_deflate (1.5.x).
- Poco consumo (menos de 1 MB).

Soporte a la aplicación

Lighttpd soporta los interfaces FastCGI, SCGI y CGI para programas externos, permitiendo que se use en este servidor aplicaciones web escritas en cualquier lenguaje. Debido a que es un lenguaje especialmente popular, se la ha dado especial atención al desarrollo de PHP. El FastCGI de Lighttpd puede ser configurado para soportar PHP con cache de códigos de operaciones tal como Alternative PHP

Cache (APC) de una manera eficaz y adecuada. Además, ha recibido bastante atención de las comunidades de Ruby on Rails y Lua debido a su popularidad. Hay que destacar en cambio que Lighttpd no soporta Microsoft's Internet Server Application Programming Interface (ISAPI).

Uso

Lighttpd esta siendo actualemnte por algunos de los sitios más grandes de Internet, incluyendo sitios como YouTube, Wikipedia y meebo, debido a su optimización para servir ficheros grandes.

5.1.2.5 Servidor WEBrick.

Las siguientes definiciones han sido tomadas de:

- http://www.webrick.org/
- http://en.wikipedia.org/wiki/WEBrick

WEBrick es una librería para servidores HTTP escrita por TAKAHASHI Masayoshi y GOTOU Yuuzou, además de los continuos parches aplicados por la inmensa comunidad de usuarios y desarrolladores de Ruby. Comenzó con un artículo titulado "Internet Programming with Ruby" en una revista de ingeniería de redes japonesa "OpenDesign"'. Y ahora, es parte de las librerías estándar desde Ruby 1.8.

Se puede utilizar la clase WEBrick de Ruby para funcionar como servidor de aplicaciones basadas en HTTP. Se puede usar también como base del desarrollo de nuevos frameworks para la creación de aplicaciones Web como por ejemplo IOWA, Tofu, etc.

También puede ser usado para crear servidores no basados en HTTP, como el servidor Daytime Server de ejemplo en la página de Inicio de WEBrick , aunque en ese caso no se tendrá la posibilidad de usar el soporte para el protocolo HTTP en ningún momento.

En el paradigma de las aplicaciones Web, WEBrick es de un nivel bastante bajo. Sin embargo se usa bastante a menudo como servidor inicial para el entrono de desarrollo o de pruebas de las aplicaciones de Ruby. Esto es debido a que su configuración es muy sencilla (un fichero para cada tecnología externa a aplicar y un solo fichero de configuración interna). En general es fiable (tiene un bug de servicio pero se está solucionando para las versiones posteriores. Versión 2.0 solucionado) y la velocidad de servicio es aceptable. Es por esto que normalmente se utiliza este servidor para desarrollo y pruebas de las aplicaciones y para producción se cambia al modelo de Mongrel, esto es un servidor Apache o lighttpd y varios Mongrel por detrás.

5.1.2.6 Decisión.

Finalmente se eligió WEBrick como servidor de la aplicación debido a las siguientes razones:

- Es el servidor más recomendado para desarrollo y pruebas de aplicaciones Web en Ruby.
- Es totalmente gratuito.
- Es open source.
- Debido a que se escogió Ruby y Ruby on Rails, las opciones de BEA Weblogic y JBOSS fueron desechadas.
- Su configuración es sencilla y debido a que este era el primer intento de programación en Ruby con los servidores relacionados con este, era el servidor más aceptable.
- La mayoría de los IDEs de desarrollo con Ruby imponen este servidor por defecto al crear proyectos permitiendo añadir servidores Mongrel de forma manual pero teniendo que configurar todos los parámetros del proyecto.

5.1.3 Framework Web (MVC).

En este punto debíamos decidir entre varios framework que nos ayudasen a conseguir un patrón Modelo –Vista-Controlador:

- **J2EE:**
  1. Struts.
  2. Spring.
- **Ruby:**
  1. Ruby On Rails.
- **PHP**
- **Python**

5.1.3.1 Struts (Java).

Las siguientes definiciones han sido tomadas de:

- http://en.wikipedia.org/wiki/Struts
- http://en.wikipedia.org/wiki/Struts
- http://struts.apache.org/

Struts es un framework de código abierto para realizar aplicaciones Web en Java bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Estas aplicaciones Web se diferencian de las aplicaciones Web escritas sólo en código HTML(.html, .htm) en que estas pueden generar respuestas dinámicas a las acciones de un usuario, comunicarse con bases de datos y con una lógica de negocio y desencadenar una respuesta.

Componentes del Modelo:

Corresponden a la lógica del negocio con la cual se comunica la aplicación Web. Usualmente el modelo comprende accesos a Bases de Datos o sistemas que funcionan independientemente de la aplicación Web.

Componentes de control:

Los componentes de control son los encargados de coordinar las actividades de la aplicación, que van desde la recepción de datos del usuario, las verificaciones de forma y la selección de un componente del modelo a ser llamado. Por su parte los componentes del modelo envían al control sus eventuales resultados o errores de manera de poder continuar con otros pasos de la aplicación.

Esta separación simplifica enormemente la escritura tanto de vistas como de componentes del modelo: Las páginas JSP no tienen que incluir manejo de errores, mientras que los elementos del control simplemente deciden sobre el paso siguiente.

Entre las características de Struts se pueden mencionar:

- Configuración del control centralizada.
- Interrelaciones entre acciones y página u otras acciones se especifican por tablas XML en lugar de codificarlas en los programas o páginas.
- Componentes de aplicación, que son el mecanismo para compartir información bidireccionalmente entre el usuario de la aplicación y las acciones del modelo.
- Librerías de entidades para facilitar la mayoría de las operaciones que generalmente realizan las páginas JSP.
- Struts contiene herramientas para validación de campos de plantillas bajo varios esquemas que van desde validaciones locales en la página (en javaScript) hasta las validaciones de fondo hechas a nivel de las acciones.
- Struts permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la vista intercomunicados por un control centralizado. Diseñando de esta manera puede obtenerse una aplicación más consistente y más fácil de mantener.

Arquitectura MVC de Struts

1.- Modelo

En Struts, el modelo se puede dividir en dos grupos:

- **Beans:** los objetos de los que consta la aplicación, los nombres a la hora de hablar de la aplicación. Normalmente, estos objetos son JavaBeans. Estas pueden ser persistentes y mantener su estado en el tiempo o sino, pueden ser conductores de la información desde otros componentes. Estos serían bases de datos, cgi, Enterprise Java Beans, servidores LDAP, etc.
- **Aciones sobre los beans:** Básicamente son el conjunto de métodos que permiten la interacción con los beans. Gets, sets, validation, etc.

2.- Vista

En Struts, la vista esta compuesta por las **JSP** (Java Server Pages), que son páginas con código HTML, a veces XML, que tiene la posibilidad de que se le puede insertar fragmentos de código embebido lo que las permite mostrar contenido dinámicamente (Actualamente con JSTL y EL ya no hay mucha necesidad de insertar fragmentos de código).

El entorno de JSP incluye también un conjunto de "tags", por ejemplo <jsp: useBean>, que se encarga de sustituir en parte al tradicional código HTML y que permite mostrar el contenido de beans, validaciones, formularios, y cualquier etiqueta o comportamiento de HTML. Además, si esto no fuera suficiente, también permite la edición de unas "custom tags", en las que podemos definir el comportamientos, parámetros de entrada, salida, etc para que la vista de la aplicación se adapte a nuestras necesidades.

3.- Controlador

Struts también proporciona el controlador de la aplicación. El controlador se centra en recibir la petición del cliente (normalmente un web browser), y decide qué lógica del negocio tiene que ser aplicada, delegando luego la responsabilidad de la siguiente fase del interfaz del usuario a un componente de la Vista. El principal componente del Controlador en el framework de Struts es el ActionServlet The primary component of the Controller in the framework is a servlet of class ActionServlet. Este servlet se configure definiendo una serie de ActionMappings. Un ActionMapping define un camino que se compara contra las URI de la petición y normalmente especifica el nombre de una clase Action. Todos los actions son subclases de org.apache.struts.action.Action. Estos Actions encapsulan las llamadas a los métodos que implementan la lógica de negocio de la aplicación y finalmente redirigen la accion al componente apropiado de la Vista para generar la respuesta al usuario.

Los Actions son básicamente las operaciones que podemos realizar sobre los objetos (Beans). Hay que diferenciar entre, lo que se puede hacer con un objeto (Action) y como hacerlo (Controller). Se asociará un Action a cada objeto de la aplicación y en este Action se definirá que comportamientos estan permitidos en el objeto. Por ejemplo, crear un nuevo objeto, editarlo, borrarlo, etc.

Además, este framework también soporta la funcionalidad de usar ActionMappings que tienen propiedades más allá de las requeridas para operar con el controlador. Esto permite almacenar información adicional como son estadísticas. También, este framework permite definir nombres lógicos con los que se controla cómo debe ser enviada una acción incluso sin saber la localización de la página JSP correspondiente. Esta funcionalidad ayuda de sobremanera a la hora de separar la lógica de control de la lógica de visión.

5.1.3.2 Spring (Java).

Las siguientes definiciones han sido inspiradas en:

- http://en.wikipedia.org/wiki/Spring_Framework
- http://www.springsource.org/

Spring, al igual que Struts, es un framework de código abierto para realizar aplicaciones Web en Java. Se le considera heredero de Struts, ya que los creadores de Spring se han valido de la experiencia de Struts para crear un framework más completo con Spring aunque tanto Struts como J2EE han seguido evolucionando. También se le considera la alternativa a las Enterprise Java Beans.

Ofrece un contenedor ligero de beans para los objetos de la capa de negocio, Data Access Objects (DAOs) y repositorio de Datasources JDBC y sesiones Hibernate. Mediante un fichero xml definimos el contexto de la aplicación siendo una potente herramienta para manejar objetos Singleton o "factorías" que necesitan su propia configuración.

Arquitectura MVC de Spring

Spring ofrece las siguientes ventajas:

- El MVC de Spring es muy flexible, ya que implementa toda su estructura mediante interfaces y no obliga a heredar de clases concretas tanto en sus Actions como en sus Forms. Evita la herencia de una clase de manera forzosa y una dependencia directa en el controller del servlet que sirve las peticiones. Proporciona una serie de implementaciones de Controllers para que el usuario los utilice. Además, todas las partes del framework son configurables via plug-in en la interface.

- Provee interceptores y controllers que permiten mantener un comportamiento común en el manejo de múltiples requests.

- No obliga a utilizar JSP, permite utilizar XLST, Velocity o implementar tu propio lenguaje para integrarlo en la View de la aplicación.

- Los controllers se configuran mediante Inversion of Control (IoC) como los demás objetos, lo cual los hace fácilmente testeables e integrables con otros objetos que estén en el contexto de Spring, y por tanto sean manejables por éste.

- La capa Web de Spring es una pequeña parte en lo alto de la capa de negocio, lo cual parece una buena práctica. Otros frameworks Web dejan a tu elección la implementación de los objetos de negocio, mientras que Spring ofrece un framework para todas las capas de la aplicación.

- No existen ActionForms sino que se enlaza directamente con los beans de negocio.

5.1.3.3 Ruby On Rails.

Las siguientes definiciones han sido inspiradas en:

- http://en.wikipedia.org/wiki/Ruby_on_Rails
- http://www.rubyonrails.org.es/

Arquitectura MVC de Rails

Las piezas de la arquitectura Modelo Vista Controlador en Ruby on Rails son las siguientes:

1.- Modelo

En las aplicaciones Web orientadas a objetos sobre bases de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos.

En Ruby on Rails, las clases del Modelo son gestionadas por ActiveRecord. Por lo general, lo único que tiene que hacer el programador es heredar de la clase ActiveRecord::Base, y el programa averiguará automáticamente qué tabla usar y qué columnas tiene. ActiveRecord esta pensado para palabras inglesas, por lo que siempre tenderá a mapear las clases (modelos) que creamos a los plurales (de los nombres de las tablas dentro de la b.b.d.d.) considerados en ingles, no en castellano. Esto aunque parezca que puede plantear problemas de mapeo, esta solventado ya que ruby on rails permite mapear cualquier clase que encontremos en el modelo a cualquier tabla de b.b.d.d. que tengamos si no queremos utilizar el mapero por defecto. También permite declarar cualquier propiedad de la clase modelo, como la Pk de una tabla.

Las definiciones de las clases también detallan las relaciones entre clases con setencias de mapeo objeto relacional:

- **has_many:** relación 1 a n. Por ejemplo, si la clase Imagen tiene una definición has_many: comentarios, y existe una instancia de Imagen llamada a, entonces a.comentarios devolverá un array con todos los objetos Comentario cuya columna imagen_id (en la tabla comentarios) sea igual a a.id.

- **belongs_to:** relación n a 1. Siguiendo en el ejemplo anterior un Comentario pertenece a Imagen. Por lo que si se accede al Comentario, siempre sabremos la Imagen a la que pertenece.

Las rutinas de validación de datos (p.e. validates_uniqueness_of:checksum) y las rutinas relacionadas con la actualización (p.e. after_destroy:borrar_archivo, before_update:actualizar_detalles) también se especifican e implementan en la clase del modelo. Al igual que en los formularios de Struts de Java, se pueden indicar reglas de validación de datos (método valídate de Struts). Si estas reglas no se cumplen, el objeto no sufrirá cambio ninguno en b.b.d.d. y se retornará al formulario de entrada de datos indicando donde se ha producido los errores y manteniendo los valores correctos activos en el formulario.

El modelo representa:

- Las Tablas de la Base de Datos.
- Migraciones (Expresan Cambios en las BD)
- Observadores

2.- Vista

En MVC, Vista es la lógica de visualización, o cómo se muestran los datos de las clases del Controlador. Con frecuencia en las aplicaciones Web la vista consiste en una cantidad mínima de código incluido en HTML.

Existen en la actualidad muchas maneras de gestionar las vistas. El método que se emplea en Rails por defecto es usar Ruby Embebido (archivos.rhtml), que son básicamente fragmentos de código HTML con algo de código en Ruby, siguiendo una sintaxis similar a JSP. También pueden construirse vistas en HTML y XML con Builder o usando el sistema de plantillas Liquid. También existen otro tipo de archivos, archivos.rxml con los que se puede procesar el formato de salida de páginas xml.

Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita mostrar información al usuario. El "maquetado" o distribución de los elementos de la página se describe separadamente de la acción del controlador y los fragmentos pueden invocarse unos a otros.

Como ayuda al desarrollo de la vista dispondremos de "Layouts" (plantillas comunes de código html que pueden ser aplicadas a cualquier rhtml de la aplicación, normalmente cabeceras comunes en la aplicación o pies de página), "Helpers" (funciones de código ruby que pueden ser accedidas desde los rhtml) y plantillas para crear la vista. Estos pueden ser invocados desde el controlador (Layouts) y afectarán a todos los archivos.rhtml que dependan de este.

3.- Controlador

En MVC, las clases del Controlador responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del Modelo y muestra los resultados usando las Vistas. En las aplicaciones Web basadas en MVC, los métodos del controlador son invocados por el usuario usando el navegador Web.

La implementación del Controlador es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Una aplicación Rails simplemente hereda de esta clase y define las acciones necesarias como métodos, que pueden ser invocados desde la Web, por lo general, una petición http de la forma "http://aplicacion/ejemplo/metodo", provoca la invocación a "EjemploController#metodo", y presenta los datos obtenidos en el método anterior usando el archivo de plantilla "/app/views/ejemplo/metodo.rhtml", a no ser que el método redirija a algún otro lugar.

Rails también proporciona andamiaje, que puede construir rápidamente la mayor parte de la lógica y vistas necesarias para realizar las operaciones más frecuentes. Esto es que con una llamada al método "scaffold" de cada controlador, no es necesario crear los métodos más comunes de cada entidad, es decir, no se deberán escribir los métodos de creación, edición y borrado de entidades de la b.b.d.d., al igual que no se necesitará crear los archivos.rhtml que contienen los formularios que realizan estas operaciones así como tampoco será necesario escribir el rhtml para listar los elementos de la b.b.d.d. Este método de andamiaje sin embargo sólo cubre el comportamiento más básico de las operaciones. Si queremos que estas operaciones realicen otras funciones además de las de crear, editar o borrar, por

ejemplo, mandar un mail cada vez que se modifica algo en b.b.d.d., será necesario sobrescribir los métodos.

Otros módulos

Además, Rails ofrece otros módulos, como Action Mailer (para enviar correo electrónico) o Active Resource que proporciona la infraestructura necesaria para crear de manera sencilla recursos REST algo por lo que apuesta claramente Rails en sus últimas versiones desplazando así a otros modelos como SOAP y XML-RPC a los que se les daba soporte en versiones anteriores mediante Action Web Service.

5.1.3.4 Decisión.

Finalmente, y para la aplicación de control de datos, se optó por Ruby on Rails. Esta decisión fue tomada conjuntamente con la de la arquitectura (Ruby) ya que no se barajaba ninguna otra posibilidad de modelo para la arquitectura seleccionada.

Igualmente, Ruby on Rails tiene numerosas ventajas como framework de desarrollo. Como se explicó antes, tiene numerosas librerías con la mayoría de los casos posibles ya implementados y modificar código en estas librerías es sumamente fácil. También es gratuito, open source y además cuenta con numerosos sitios Web que respaldan a los desarrolladores (RubyForge) por lo que encontrar ayuda a problemas o soporte es sencillo también. Además de esto, Ruby on Rails cuenta con la ventaja de que que es un framework en el que se desarrolla muy rápido (gracias a la metaprogramación, scripts y el propio lenguaje Ruby). Ruby on Rails también se trata de una tecnología relativamente novedosa pero que ya esta lo suficientemente asentada como para que su uso esté ampliamente extendido y se hayan eliminado muchos de los problemas que dan las tecnologías tan novedosas.

En este caso la razón de más peso por la que se decidió usar Ruby on Rails es por establecer una línea divisoria entre los dos proyectos que comparte la aplicación. Por lo que el director de proyecto sugirió establecer este modelo para la aplicación de control de datos. Además, al alumno que llevaba la parte de presentación se le ofreció el proyecto a desarrollar en Java por lo que no había razón para modificarle esta oferta.

5.1.4 Base de Datos.

A la hora de seleccionar la base de datos, tendremos en cuenta sólo las versiones gratuitas más conocidas y robustas del mercado.

5.1.4.1 SQL Server.

Las siguientes definiciones han sido inspiradas en:

- http://es.wikipedia.org/wiki/SQL_Server:
- http://www.microsoft.com/spain/sql/default.mspx

Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Las características de SQL Server están organizadas en las siguientes 3 áreas:

Plataforma Confiable: mayores niveles de **seguridad**, **confiabilidad** y **escalabilidad** para aplicaciones de misión crítica.

- **Protección de información valiosa (Seguridad)**:
  - Encriptación transparente de datos.
  - Administración extensible de llaves.
  - Auditoría.
- **Asegurar continuidad de negocios (Alta Disponibilidad)**:
  - Espejo (Mirror) de base de datos mejorado.
  - Recuperación automática de páginas de datos.
  - Compresión del tráfico de Logs.
- **Habilitar una respuesta predecible (Escalabilidad)**:
  - Resource Governor.
  - Rendimiento de Consultas predecible.
  - Compresión de datos.
  - Adición de CPU en "caliente".

Plataforma Productiva: permite crear e implementar soluciones de negocios basadas en datos de forma rápida reduciendo así el costo de administración y desarrollo de aplicaciones.

- **Administración por políticas (Administración)**:
  - Administración basada en políticas.
  - Instalación integrada.
  - Recolección de datos de rendimiento.
- **Simplificar el desarrollo de aplicaciones**:
  - Lenguaje Integrado de Consultas (LINQ).

- Servicio de objetos en ADO.NET.

- **Almacenar cualquier información**
  - DATE/TIME.
  - HIERARCHY ID.
  - Datos FILESTREAM.
  - Búsqueda Full-text integrada.
  - Sparse Columns: son columnas normales que han sido optimizadas para el almacenamiento de valores nulos.
  - Tipos de datos de usuario Grandes (BLOBs, CLOBs).
  - Tipos de datos Espaciales(Spatial Data).

Plataforma Inteligente: es una plataforma comprehensiva que permite entregar inteligencia donde el usuario lo necesita.

- **Integrar cualquier dato**:
  - Compresión de respaldos de datos.
  - Paralelismo en Tablas particionadas.
  - Optimizaciones en consultas Star Join.
  - Grouping Sets.
  - Captura en datos con cambios.
  - Sentencia SQL MERGE.
  - SQL Server Integration Services (SSIS) con mejoras en la línea de datos.
  - SQL Server Integration Services (SSIS) búsquedas persistentes.
  - Datawarehousing.
- **Entregar información relevante**
  - Análisis escalable y de buen rendimiento.
  - Cómputos por bloque.
  - Writeback.
- **Encontrar ideas accionables**
  - Motor de reportes empresarial.
  - Implementación de reportes en Internet.
  - Administración de infraestructura de reportes.
  - Mejoras en la construcción de reportes.
  - Autenticación por FORMS integrada.
  - Aplicaciones del servidor de reportes embebidas.
  - Integración con Microsoft Office.
  - Análisis Predictivo.
  - Servicio de Reportes.

5.1.4.2 PostgreSQL.

Las siguientes definiciones han sido inspiradas en:

- http://es.wikipedia.org/wiki/PostgreSQL/:
- http://www.postgresql.org/about/

PostgreSQL es un servidor de base de datos relacional orientado a objetos de software libre, liberado bajo la licencia BSD que ha sido desarrollado en diferentes formatos desde 1977. Comenzó como un proyecto llamado Ingres en la Universidad de California en Berkeley. Ingres sería luego desarrollado comercialmente por "Relational Technologies/Ingres Corporation".

En 1986 otro equipo de Berkeley dirigido por Michael Stonebraker continuo el desarrollo del código de Ingres para crear un servidor de base de datos relacional orientado a objetos llamado Postgres. En 1996, gracias a un nuevo esfuerzo para maximizar la funcionalidad y establecer un "open source" del software, Postgres se renombró a PostgreSQL, después de un pequeño periodo como Postgres95. El proyecto PostgreSQL está todavía en desarrollo bajo un grupo mundial de desarrolladores "open source" y contribuidores.

PostgreSQL es considerado por la gran mayoría como una de las bases de datos "open source" más avanzadas. Proporciona numerosas funcionalidades que normalmente están sólo presentes en los productos comerciales.

Versión Gratuita "Open Source"

PostgreSQL es un proyecto "open source". Open source por definición significa que el código fuente puede ser revisado en cualquier momento, usar el programa y modificarlo libremente sin las limitaciones presentes en el software propietario. Referido a las bases de datos, open source significa que tienes acceso a las estadísticas, que otras compañías prohíben.

Sin embargo, hay una equivocación general por la que se considera que por ser una distribución de software open source libre de restricciones, esta va a ser gratuita para la empresa que lo utilice. Este no es necesariamente el caso. Es cierto que se puede descargar e instalar el software sin coste alguno (en principio, la versión básica. En caso de que se requiera de funcionalidades más avanzadas o soporte contínuo es posible que se requiera de alguna inversión extra), pero siempre habrá un coste alternativo en el soporte y adaptación de la aplicación.

Características principales:

- **DBMS Objeto-Relacional**
  PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arrays.

- **Altamente_Extensible**
  PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

- **Soporte_SQL_Comprensivo**
  PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como la operación 'union join' de SQL92.
- **Integridad Referencial**
  PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

- **API Flexible**
  La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfacaes incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

- **Lenguajes Procesuales**
  PostgreSQL tiene soporte para lenguajes procesuales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al "procedural language" (lenguaje que se ejecuta línea a línea de código) de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procesual embebido.

- **MVCC**
  MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control). PostgreSQL controla la concurrencia mediante este sistema. Este sistema le da a cada usuario una fotografía de la BBDD en un momento dado, permitiendo que se hagan cambios en esta sin que sean visibles para el resto de usuarios hasta que se realiza un "commit" de la transacción. Esto elimina por completo la necesidad de "locks" y asegura que la BBDD mantiene los principios ACID (Atomicity, Consistency, Isolation, Durability) de una manera eficiente.

  Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

  En relación con los estándares ANSI/ISO SQL (Serializable, Repeatible Read, Read Committed, Read Uncommitted) PostgreSQL no garantiza que el orden en que se ejecutan las transacciones seguirán el mismo orden a la salida. Esto contradice explícitamente el standard ANSI/ISO SQL 99.

- **Cliente/Servidor**
  PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

- **Write Ahead Logging (WAL)**
  La característica de PostgreSQL conocida como *Write Ahead Logging* fuerza a que se escriban las operaciones en el registro de cambios antes de que estos sean escritos en la base de datos.

Esto garantiza que existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser muy beneficioso ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado.

5.1.4.3 MySQL Server.

Las siguientes definiciones han sido tomadas de:

- http://es.wikipedia.org/wiki/MySQL/:
- http://www.mysql.com/

MySQL es la base de datos open source más popular. Su continuo desarrollo y su creciente popularidad están haciendo de MySQL un competidor cada vez más directo de gigantes en la materia de las bases de datos como Oracle.

MySQL es un sistema de administración de bases de datos (*Database Management System*, *DBMS*) para bases de datos relacionales.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza multiples tablas para almacenar y organizar la información.

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su ínteractuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente (salvo aquellas en las que se ofrece soporte dedicado). Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

Características de la versión 5.0.22

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multiplataforma.
- Procedimientos almacenados.
- Triggers.
- Cursores.
- Vistas actualizables.
- Soporte a VARCHAR.
- INFORMATION_SCHEMA.
- Modo Strict.
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle.
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial).
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación (savepoints) con InnoDB.
- Soporte para SSL.
- Query caching.
- Sub-SELECTs (o SELECTs anidados).
- Réplica con un maestro por esclavo, varios esclavos por maestro, sin soporte automático para multiples maestros por esclavo.
- indexing y buscando campos de texto completos usando el motor de almacenamiento MyISAM.
- Embedded database library.
- Soporte completo para Unicode.
- Conforme a las reglas ACID usando los motores InnoDB, BDB y Cluster.
- Shared-nothing clustering through MySQL Cluster.

La gran mayoría de gente usa este gestor en Internet por sus ventajas:

- Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
- Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
- Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.
- El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro y de buscadores.

5.1.4.4 Decisión.

Como comparación entre MySQL, Postgres y SQLServer, parece aceptado que MySQL junto con un servidor HTTTP serían suficientes para servir páginas web con contenido dinámico, discusiones, noticias, etc. En general, sistemas en los que la velocidad y el número de accesos concurrentes sea algo primordial, y la seguridad no sea muy importante (pueda bastar con hacer backups periódicos que se restaurarán tras una caida del servidor). En cambio, para sistemas más serios en las que la consistencia de la BD sea fundamental (bancos, etc.) PostgreSQL sería una mejor opción pese a su mayor lentitud. Respecto a SQLServer, debido a los costes que supone su implantación debemos rechazar la idea de aplicarlo al desarrollo de este proyecto.

Finalmente la decisión que se tomó fue la de seleccionar MySQL como base de datos debido a:

- Las ventajas nombradas anteriormente: Velocidad, fiabilidad, bajo consumo de este servidor de b.b.d.d.

- MySQL es gratuito y open source lo que permite modificarlo para adaptarlo a las necesidades de nuestra aplicación.

- Los dos alumnos que participamos en la aplicación estábamos más familiarizados con este servidor y con su entorno de configuración y administración.

5.2 Resumen.

Para esta iteración de la aplicación se han decidido las siguientes tecnologías basándonos principalemente en el guión del PFC, en el precio y mantenimiento de las tecnologías y en la familiaridad que tenía el proyectando con ellas:

- **ARQUITECTURA:**
  - Ruby on Rails (Ruby).

- **SERVIDOR WEB:**
  - WEBRick Server.

- **FRAMEWORK:**
  - Ruby on Rails.

- **DASE DE DATOS:**
  - MySQL

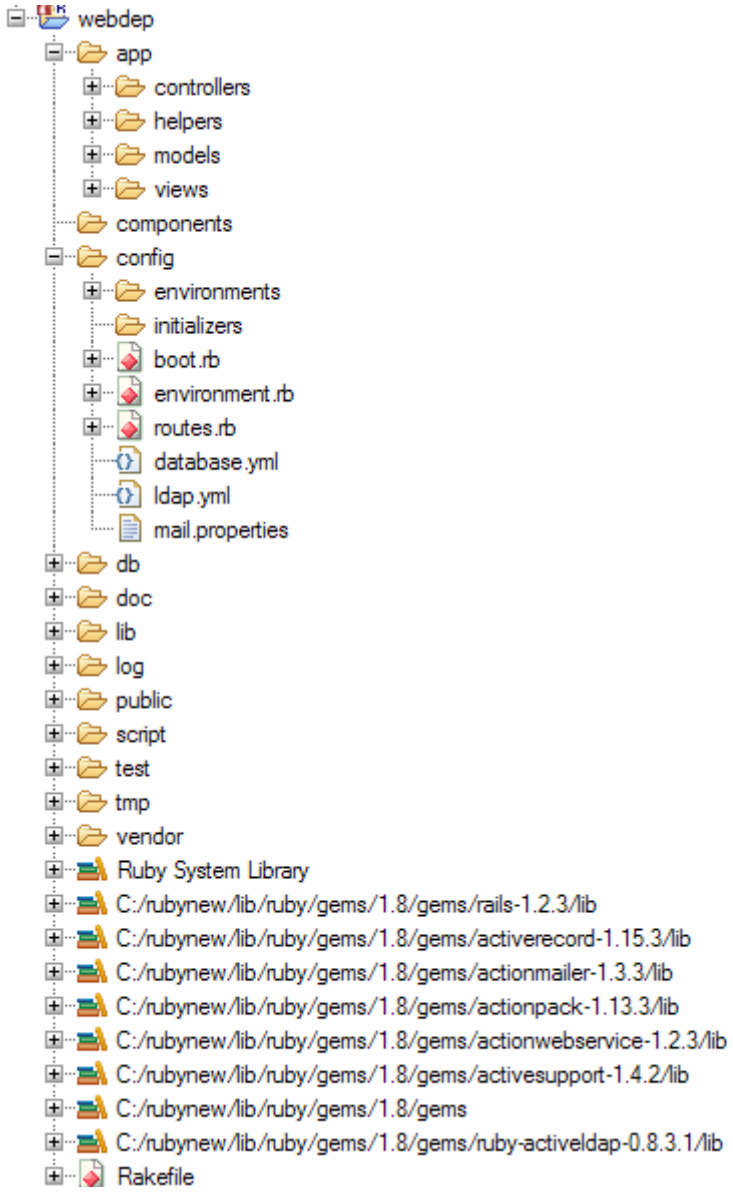
En furutas iteraciones de la aplicación se puede contemplar el modificar algunas de las tecnologías elegidas en esta ocasión. Por ejemplo: Servidor de Aplicaciones. Ya que si se planea que la aplicación sea accedida por numerosos usuarios sería recomendable seguir otro tipo de arquitectura de red. En puntos posteriores de esta memoria se explican las posibles mejoras o cambios que se podrían realizar sin que ello suponga cambio alguno en la estructura de la aplicación o en el propio código.

6. Descripción de paquetes del proyecto.

A continuación se hará una descripción de la estructura y los paquetes de la aplicación, indicando en cada fichero su funcionalidad dentro de la aplicación.

Hay que destacar en la figura los siguientes directorios:

- App: Contiene todos los archivos referentes a la aplicación.
- Config: Contiene toda la configuración del servidor de la aplicación
- Lib: Contiene todos aquellos paquetes externos al desarrollo del proyecto.



6.1 Directorio de Aplicaciones.

App: Para seguir el modelo MVC, los ficheros de la aplicación se subdividen en varias partes:

- Controllers: Contiene todos los controladores de la aplicación, es decir, el comportamiento lógico. Existe un controlador por cada uno de los modelos de la aplicación así como otros controladores extra para que permiten redirigir el paginado y las acciones de la aplicación.

- Helpers: Contienen las clases que ayudan a los controladores. Por ejemplo una clase de utilidades que se vaya a usar en varios controladores. Por defecto, cuando se crea un Controller, también se crea un Helper. En este desarrollo no hemos necesitado completar ningún Helper.

- Models: Contiene los ficheros que referencian a las entidades presentes en b.b.d.d.

- Views: Contiene las páginas rhtml de la aplicación.

6.1.1 Controladores.

A continuación se describirán los Controllers de la aplicación y los métodos que los componen:

- **Account_Controller:**
  - delete: Se encarga del "LOG OUT" de la aplicación. Elimina la sesión del usuario actual.

- **Application:**
  - autentícate: Crea la sesión para el usuario. En caso de error redirige al login.

  - roleAdmin: Filtro que determina si el perfil del usuario es Administrador. En caso contrario redirige al login.

  - filterEmpleado: Filtro que determina si el perfil del usuario es Empleado. En caso contrario redirige al login.

  - filterAdministrativo: Filtro que determina si el perfil del usuario es Administrativo. En caso contrario redirige al login.

  - filterTecnico: Filtro que determina si el perfil del usuario es Tecnico. En caso contrario redirige al login.

  - Paginate_Collection: Método que permite paginación de las tablas de datos de la aplicación.

  - Load_properties: Método que permite la carga de los datos de un fichero *.properties en un Hash para su posterior lectura.

- System: Método que permite hacer llamadas al sistema.

- **Asignatura_Controller:**
  - new: Carga los datos de la página para poder crear una nueva asignatura.
  - List: Carga los datos de todas las asignaturas para poder presentarlas.
  - Edit: Carga los datos de una asignatura en concreto y todos los datos necesarios para modificarla.
  - Create: Inserta en b.b.d.d la asignatura.
  - checkImparte: Comprueba que hay al menos un empleado autorizado impartiendo la asignatura.
  - Imparte: Carga los datos de la página para asignar los empleados que imparten la asignatura.
  - Asignar: Asigna el empleado seleccionado a la asignatura.
  - Desasignar: Elimina al empleado de la lista de empleados que imparten la asignatura.
  - Pertenece: Carga el listado con las titulaciones a las que pertenece la asignatura.
  - asignarTitulacion: asigna la asignatura a la Titulación escogida.
  - desasignarTitulacion: elimina la asignatura de la Titulación escogida.
  - Update: Accede a b.b.d.d y modifica la asignatura con los datos proporcionados.
  - DeleteAsignatura: Elimina la asignatura y todos los datos dependientes de ella.

- **Campus_Controller:**
  - new: Carga los datos de la página para poder crear un campus nuevo.
  - List: Carga los datos de todos los campus para poder presentarlos.
  - Edit: Carga los datos de un campus en concreto y todos los datos necesarios para modificarlo.
  - Create: Inserta en b.b.d.d el campus.
  - Update: Accede a b.b.d.d y modifica el campus con los datos proporcionados.

- DeleteCampus: Elimina el campus y todos los datos dependientes de él.

- **Despacho_Controller:**
  - new: Carga los datos de la página para poder crear un despacho nuevo.
  - List: Carga los datos de todos los despachos para poder presentarlos.
  - Edit: Carga los datos de un despacho en concreto y todos los datos necesarios para modificarlo.
  - Create: Inserta en b.b.d.d el despacho.
  - Update: Accede a b.b.d.d y modifica el despacho con los datos proporcionados.
  - DeleteDespacho: Elimina el despacho y todos los datos dependientes de él.
  - UpdateDespacho: Comprueba que cuando se quiere modificar el despacho de un empleado, éste cuente al menos con un despacho asignado.
  - AsignarDespacho: Asigna el despacho al empleado seleccionado.

- **Empleado_Controller:**
  - new: Carga los datos de la página para poder crear un empleado nuevo.
  - List: Carga los datos de todos los empleados para poder presentarlos.
  - Edit: Carga los datos de un empleado en concreto y todos los datos necesarios para modificarlo.
  - Create: Inserta en b.b.d.d el empleado.
  - Despacho: Carga los datos para asignar un despacho al empleado.
  - AsignarDespacho: Asigna el despacho seleccionado el empleado.
  - DespachoAsigned: Comprueba que el empleado tiene al menos un despacho asignado.
  - DesasignarDespacho: Elimina el despacho seleccionado del lista de despachos del empleado.
  - Update: Accede a b.b.d.d y modifica el empleado con los datos proporcionados.
  - DeleteEmpleado: Elimina el empleado y todos los datos dependientes de él.
  - UpdateAsignatura: Carga los datos de asignaturas del empleado para modificarlos.

- UpdateAutorTesis: Carga los datos de las tesis de las que es autor el empleado para modificarlos.

- UpdateDirectorTesis: Carga los datos de las tesis de las que es director el empleado para modificarlos.

- UpdateCoordinadorProyecto: Carga los datos de los proyectos de los que es coordinador el empleado para modificarlos.

- UpdateResponsableProyecto: Carga los datos de los proyectos de los que es responsable el empleado para modificarlos.

- UpdatePublicacion: Carga los datos de las publicaciones del empleado para modificarlos.

- UpdateEmpleadoAsignatura: Modifica las asignaturas del empleado seleccionado.

- UpdateEmpleadoAutorTesis: Modifica las tesis de las que es autor el empleado seleccionado.

- UpdateEmpleadoDirectorTesis: Modifica las tesis de las que es director el empleado seleccionado.

- UpdateEmpleadoCoordinadorProyecto: Modifica los proyectos de los que es coordinador el empleado seleccionado.

- UpdateEmpleadoResponsableProyecto: Modifica los proyectos de los que es responsable el empleado seleccionado.

- UpdateEmpleadoPublicacion: Modifica las publicaciones de las que es autor el empleado seleccionado.

- Empleadoml: Obtiene los datos del empleado y crea un xml para mostrarlo en pantalla.

- **Error_Controller:**
  - error: Método que redirecciona a la pantalla de error mostrando un error determinado.

- **Evento_Controller:**
  - new: Carga los datos de la página para poder crear un evento nuevo.

  - List: Carga los datos de todos los eventos para poder presentarlos.

  - Edit: Carga los datos de un evento en concreto y todos los datos necesarios para modificarlo.

- Create: Inserta en b.b.d.d el evento.

- Update: Accede a b.b.d.d y modifica el evento con los datos proporcionados.

- DeleteEvento: Elimina el evento y todos los datos dependientes de él.

- AnunciaGrupo: Carga los datos para poder anunciar el evento en un grupo.

- AsignarGrupo: Asigna el evento al grupo seleccionado.

- DesasignarGrupo: Elimina el evento del listado de eventos del grupo.

- AsignarProyecto: Asigna el evento a un proyecto de investigación.

- DesasignarProyecto: Elimina el evento del listado de eventos del proyecto de investigación.

- **Grupo_Controller:**
  - new: Carga los datos de la página para poder crear un grupo nuevo.

  - List: Carga los datos de todos los grupo para poder presentarlos.

  - Edit: Carga los datos de un grupo en concreto y todos los datos necesarios para modificarlo.

  - Create: Inserta en b.b.d.d el grupo. Cuando se crea un nuevo grupo, se inserta también en b.b.d.d una línea de investigación por defecto para este grupo con los mismo datos que éste.

  - Update: Accede a b.b.d.d y modifica el grupo con los datos proporcionados.

  - DeleteLinea: Elimina la línea de investigación asociada que ha sido seleccionada.

  - DeleteGrupo: Elimina el grupo y todos los datos dependientes de él.

  - UpdateEmpleado: Carga los datos del empleado asociado al grupo.

  - UpdateGrupoEmpleado: Modifica el grupo al que está asociado el empleado.

  - UpdateLinea: Carga los datos de la línea de investigación asociada al grupo.

  - UpdateGrupoLinea: Modifica el grupo al que está asociada la línea de investigación. Este método no es aplicable a la línea por defecto de un grupo.

- **Home_Dpto_Controller:**
  - new: Carga los datos de la página para poder crear una nueva Home de departamento.
  - List: Carga los datos de todas las Homes para poder presentarlas.
  - Edit: Carga los datos de una Homes en concreto y todos los datos necesarios para modificarla.
  - Create: Inserta en b.b.d.d la Home.
  - Update: Accede a b.b.d.d y modifica la Home con los datos proporcionados.
  - DeleteHome: Elimina la Home y todos los datos dependientes de ella.

- **Inicio_Controller:**
  - Index: Método que redirecciona al menú principal del usuario dependiendo del perfil de éste.

- **Linea_Controller:**
  - new: Carga los datos de la página para poder crear una nueva línea de investigación.
  - List: Carga los datos de todas las línea de investigación para poder presentarlas.
  - Edit: Carga los datos de una línea de investigación en concreto y todos los datos necesarios para modificarla.
  - Create: Inserta en b.b.d.d la línea de investigación.
  - Update: Accede a b.b.d.d y modifica la línea de investigación con los datos proporcionados.
  - DeleteLinea: Elimina la línea de investigación y todos los datos dependientes de ella.

- **Login_Controller:**
  - Login: Método que verifica la validez del usuario, lo identifica y lo redirige a su página de inicio.
  - Logout: Método que cierra la sesión del usuario.

- **Protected_Controller:**
  Este controller no posee métodos propios sino que hace referencia al método de autenticación del Controller application.rb.

- **Proyecto_Controller:**
  - new: Carga los datos de la página para poder crear un nuevo proyecto de investigación.
  - List: Carga los datos de todos los proyecto de investigación para poder presentarlos.
  - Edit: Carga los datos de un proyecto de investigación en concreto y todos los datos necesarios para modificarlo.
  - Create: Inserta en b.b.d.d el proyecto de investigación.
  - Update: Accede a b.b.d.d y modifica el proyecto de investigación con los datos proporcionados.
  - Trabaja: Carga los datos de todos los empleados que trabajan en el proyecto de investigación.
  - Asignar: Asigna al empleado seleccionado como trabajador del proyecto de investigación.
  - Desasignar: Elimina al empleado seleccionado del listado de empleados que trabajan en el proyecto de investigación.
  - Lineas: Carga la información de la líneas de investigación que sigue el proyecto.
  - Participa: Carga la información de las líneas en las que el proyecto no participa para mostrarlas en un listado.
  - AsignarLinea: Asigna el proyecto a la línea de investigación seleccionada.
  - DesasignarLinea: Elimina la línea de investigación del las líneas que sigue el proyecto.
  - DeleteProyecto: Elimina el proyecto de investigación y todos los datos dependientes de él.

- **Publicación_Controller:**
  - new: Redirige a la pantalla de carga de ficheros para poder crear las nuevas publicaciones.
  - UploadFile: Recoge el fichero subido, lo parsea y crea las publicaciones incluidas en el.
  - Además, presenta este listado para que el usuario modifique manualmente los autores de éstas.

- List: Carga los datos de todas las publicaciones para poder presentarlas.
- Edit: Carga los datos de una publicación en concreto y todos los datos necesarios para modificarla.
- Create: Inserta en b.b.d.d la publicación.
- Update: Accede a b.b.d.d y modifica la publicación con los datos proporcionados.
- Autor: Carga los datos de todos los autores de la publicación seleccionada.
- AsignarAutor: Asigna al empleado seleccionado como autor de la publicación.
- DesasignarAutor: Elimina al empleado del listado de autores de la publicación.
- DeletePublicación: Elimina la publicación y todos los datos dependientes de ella.

- **Tesis_Controller:**
  - new: Carga los datos de la página para poder crear una nueva tesis.
  - List: Carga los datos de todas las tesis para poder presentarlas.
  - Edit: Carga los datos de una tesis en concreto y todos los datos necesarios para modificarla.
  - Create: Inserta en b.b.d.d la tesis.
  - Update: Accede a b.b.d.d y modifica la tesis con los datos proporcionados.
  - Destroy: Elimina la tesis y todos los datos dependientes de ella.

- **Tipos_Cargo_controller:**
  - new: Carga los datos de la página para poder crear un nuevo tipo de cargo.
  - List: Carga los datos de todos los tipos de cargo para poder presentarlos.
  - Edit: Carga los datos de un tipo de cargo en concreto y todos los datos necesarios para modificarlo.
  - Create: Inserta en b.b.d.d el tipo de cargo.
  - Update: Accede a b.b.d.d y modifica el tipo de cargo con los datos proporcionados.
  - DestroyCargo: Elimina el tipo de cargo y todos los datos dependientes de él.

- UpdateCargo: Carga los datos del empleado que posee dicho cargo para actualizarlo.

- UpdateCargoEmpleado: Carga los datos de todos los empleados que poseen dicho cargo. Si no encuentra ninguno, elimina el tipo de cargo.

- **Tipos_Contrato_Controller:**
  - new: Carga los datos de la página para poder crear un nuevo tipo de contrato.

  - List: Carga los datos de todos los tipos de contrato para poder presentarlos.

  - Edit: Carga los datos de un tipo de contrato en concreto y todos los datos necesarios para modificarlo.

  - Create: Inserta en b.b.d.d el tipo de contrato.

  - Update: Accede a b.b.d.d y modifica el tipo de contrato con los datos proporcionados.

  - DeleteContrato: Elimina el tipo de contrato y todos los datos dependientes de él.

  - UpdateContrato: Carga los datos del empleado que posee dicho contrato para actualizarlo.

  - UpdateContratoEmpleado: Carga los datos de todos los empleados que poseen dicho contrato. Si no encuentra ninguno, elimina el tipo de contrato.

- **Tipos_Evento_Controller:**
  - new: Carga los datos de la página para poder crear un nuevo tipo de evento.

  - List: Carga los datos de todos los tipos de evento para poder presentarlos.

  - Edit: Carga los datos de un tipo de evento en concreto y todos los datos necesarios para modificarlo.

  - Create: Inserta en b.b.d.d el tipo de evento.

  - Update: Accede a b.b.d.d y modifica el tipo de evento con los datos proporcionados.

  - DeleteTipoEvento: Elimina el tipo de evento y todos los datos dependientes de él.

- **Tipos_Perfil_Controller:**
  - new: Carga los datos de la página para poder crear un nuevo tipo de perfil.

  - List: Carga los datos de todos los tipos de perfil para poder presentarlos.

  - Edit: Carga los datos de un tipo de perfil en concreto y todos los datos necesarios para modificarlo.

- Create: Inserta en b.b.d.d el tipo de perfil.

- Update: Accede a b.b.d.d y modifica el tipo de perfil con los datos proporcionados.

- DeletePerfil: Elimina el tipo de perfil y todos los datos dependientes de él.

- UpdatePerfil: Carga los datos del empleado que posee dicho perfil para actualizarlo.

- UpdatePerfilEmpleado: Carga los datos de todos los empleados que poseen dicho perfil. Si no encuentra ninguno, elimina el tipo de perfil.

- **Tipos_Proyecto_Controller:**
  - new: Carga los datos de la página para poder crear un nuevo tipo de proyecto.

  - List: Carga los datos de todos los tipos de proyecto para poder presentarlos.

  - Edit: Carga los datos de un tipo de proyecto en concreto y todos los datos necesarios para modificarlo.

  - Create: Inserta en b.b.d.d el tipo de proyecto.

  - Update: Accede a b.b.d.d y modifica el tipo de proyecto con los datos proporcionados.

  - DeleteTipo: Elimina el tipo de proyecto y todos los datos dependientes de él.

  - UpdateTipo: Carga los datos del proyecto que posee dicho tipo para actualizarlo.

  - UpdateTipoProyecto: Carga los datos de todos los proyectos que poseen dicho tipo. Si no encuentra ninguno, elimina el tipo de proyecto.

- **Titulacion_Controller:**
  - new: Carga los datos de la página para poder crear una nueva titulación.

  - List: Carga los datos de una titulación para poder presentarlos.

  - Edit: Carga los datos de una titulación en concreto y todos los datos necesarios para modificarla.

  - Create: Inserta en b.b.d.d la titulación.

  - Update: Accede a b.b.d.d y modifica la titulación con los datos proporcionados.

  - DeleteTitulacion: Elimina la titulación y todos los datos que dependen de ella si no tiene asociada ninguna asignatura.

- unbindAsignatura: Elimina la asignatura seccionada del listado de asignaturas de la titulación.

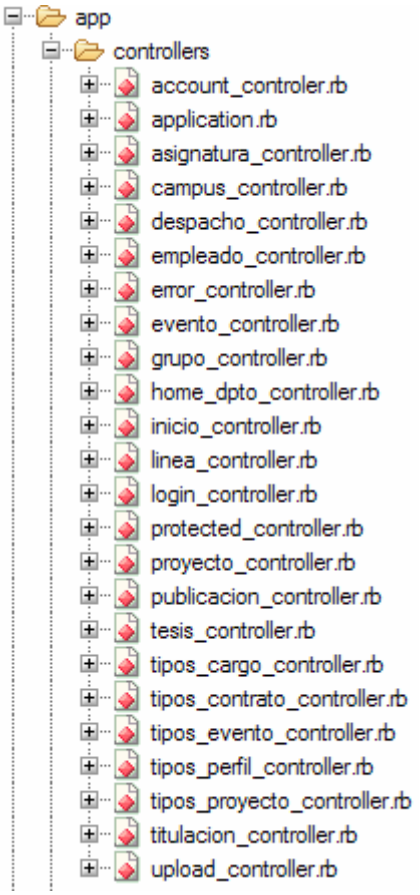
- pertenece: Carga el listado con los datos de las asignaturas que pertenecen a la titulación.

- AsignarAsignatura: Inserta la asignatura seleccionada en el listado de asignaturas que se imparten en la titulación.

- DesasignarAsignatura: Elimina la asignatura del listado de asignaturas que se imparten en la titulación.

- **Upload_Controller:**
  - index: Redirige la aplicación a la página de subida de ficheros de publicaciones.

  - uploadFile: Carga el fichero subido en b.b.d.d. con todas las publicaciones que aparecen.



6.1.2 Helpers.

Se han necesitado implementar sólo algunos métodos extra en los Helpers. Sin embargo, se crean por defecto cuando creamos un Controller.

En caso de que se necesiten crear estos métodos comunes de la aplicación, los Helpers estarán disponibles en el directorio app/helpers.

Para esta aplicación, tan sólo se han implementado métodos para comprobar en los listados de proyectos y tesis, si el usuario logado es coordinador de los proyectos mostrados o si es autor local de las tesis que se muestran.

6.1.3 Modelos.

Representan todas las entidades que podemos encontrar en la aplicación. En ellas se establecen las propiedades de las entidades, es decir, se define las relaciones entre entidades, y el valor de las propiedades como por ejemplo si son campos numéricos o alfanuméricos, si se permite nulo, etc.
Por defecto se han implementado los modelos indicando la tabla de base de datos de la que provienen además de sus calves primarias.

En algunos casos y para "engañar" a la base de datos, encontraremos métodos definidos dentro de los modelos. Estos métodos reescriben los métodos propios de inserción, actualización y borrado dentro de base de datos. En concreto se sobrescriben los métodos para poder modificar campos de las entidades que pertenecen a sus claves primarias. Esto no debería ser permitido por la base de datos y es por ello que se sobrescriben estos métodos (comentaremos este problema y su posible solución en el apartado de mejoras de la aplicación).

- **Anuncia_Grupo**
  - Indica la relación entre Eventos y Grupos.
  - `belongs_to`:Grupos (Puede tener un Grupo).
  - `belongs_to`:Eventos (Puede tener un Evento).

- **Anuncia_Proyecto**
  - Indica la relación entre Eventos y Proyectos de Investigación.
  - `belongs_to`:Proyectos (Puede tener un Proyecto).
  - `belongs_to`:Eventos (Puede tener un Evento).

- **Asignatura**
  - Representa la entidad de Asignatura de b.b.d.d.
  - `belongs_to`:Empleado (tiene un Empleado-Coordiandor).
  - `has_many`:Imparte (n relaciones de Imparte con Asignatura).
  - `validates`:
    - campos numéricos: Curso, Cuatrimestre.
    - uniqueness: URL_Uni_Es, URL_Uni_En, URL_Dpto_Es, URL_Dpto_Es.
    - presence: Titulo, Curso, Cuatrimestre, Coordinador.
  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Campus**
  - Representa la entidad de Campus de b.b.d.d.
  - `has_many`:Despachos (n relaciones de Despachos con Campus).

- **Data_file**
  - Este model no representa una entidad de b.b.d.d. sino un modelo físico, el fichero que los usuarios de la aplicación tienen subir para poder crear Publicaciones. Sobre este modelo se han creado varios métodos para acceder, parsear y eliminar el fichero subido.
  - Métodos:
    - Save: método que guarda en un directorio local el fichero subido por el usuario, lo parsea y devuelve un hash con las publicaciones encontradas para operar con él.
    - Sanitize_Filename: Método que parsea el nombre del fichero subido por el usuario.
    - Cleanup: Método que borra el fichero subido por el usuario.
    - SaveBibTexml: Método que recorre el arbol xml y recupera los datos y crea un hash que se retorna para crear las Publicaciones.

- **Despacho**
  - Representa la entidad de Despacho de b.b.d.d.
  - `has_many`:Ocupa (n relaciones de Ocupa con Despacho).
  - `belongs_to`:Campus (Puede tener un Campus; de hecho , es obligatorio).

- **Empleado**
  - Representa la entidad de Empleado de b.b.d.d.
  - `has_many`:Asignaturas (n relaciones de coordinador con Asignaturas).
  - `has_many`:Proyectos (n relaciones responsable/coordinador con Proyectos).
  - `has_many`:Publicaciones (n relaciones de responsable con Publicaciones).
  - `has_many`:Tesis (n relaciones de director con Tesis)
  - `has_many`:Es_Autor (n relaciones de Es_Autor con Publicaciones).
  - `has_many`:Imparte (n relaciones de Imparte con Asignaturas)
  - `has_many`:Ocupa (n relaciones de Ocupa con Despachos).
  - `has_many`:Trabaja_En (n relaciones de Trabaja_En en Proyectos)
  - `belongs_to`:Tipos_Cargo (puede tener un Cargo).
  - `belongs_to`:Tipos_Contrato (puede tener un Contrato; de hecho, es obligado).
  - `belongs_to`:Grupo (puede tener un Grupo).
  - `belongs_to`:Tipos_Perfil (puede tener un Perfil; de hecho, es obligado).
  - `validates`:

  - Uniqueness: Email, Login, Página URL.
  - presence: Primer Nombre, Primer Apellido, Fecha Inicio, Contrato, Perfil.

- **Es_Autor**
  - Indica la relación entre Empleados y Publicaciones de las que es autor.
  - `belongs_to`:Empleados (Puede tener un Empleado).
  - `belongs_to`:Publicaciones (Puede tener una Publicación).

- **Evento**
  - Representa la entidad Evento dentro de b.b.d.d.
  - `belongs_to`:Tipos_Evento (Puede tener un Tipo de evento; de hecho, es obligatorio).
  - `has_many`:anuncia_grupo (n relaciones de Anuncia_Grupo con grupos).
  - `has_many`:anuncia_proyecto(n relaciones de Anuncia_Proyecto con Proyectos)
  - `validates`:
    - Uniqueness: Descripción y Desciption.
    - presence: Título, URL_Es, Título_Abreviado, Fecha, Categoría.

- **Grupo**
  - Representa la entidad Grupos de b.b.d.d.
  - `has_many`:Empleados (n relaciones de grupo con Empleados).
  - `has_many`:Lineas (n relaciones de grupo con Líneas).
  - `has_many`:Eventos (n relaciones de grupo con Eventos).
  - `validates`:
    - Uniqueness: Título, Title, Abbreviated_Title,URL_Logo.
    - presence:Titulo,Email.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Home_Dpto**
  - Representa la entidad Home_Dpto de b.b.d.d.
  - `validates`:
    - presence:Email,Dirección,Nombre.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.

  - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
  - Update: método que modifica la clave primaria de una entidad ya creada.

- **Imparte**
  - Indica la relación entre Asignaturas y Empleados que imparten esas Asignaturas.
  - `belongs_to`:Empleados (Puede tener un Empleado)
  - `belongs_to`:Asignaturas (Puede tener una Asignatura)

- **Ldap_User**
  - Este model no representa una entidad de b.b.d.d. sino que representa las operaciones de autenticación sobre un usuario contra un servidor.
  - Métodos:
    - Login: método que contrasta la identidad del usuario contra una base de datos Ldap. En esta versión está contrastando la identidad contra la propia base de datos de la aplicación.
    - `authenticate`: Método que autentica al usuario contra el servidor Ldap.
    - Ldap_search: Método que busca una entrada en el directorio Ldap con los datos del usuario.
    - Login2: Método que realiza el login del usuario contra el servidor Ldap. Este es un método altenativo según la versión de Ldap utilizada.

- **Linea**
  - Representa la entidad Línea de Investigación de b.b.d.d.
  - `has_many`:Participa_En (n relaciones de lineas con Proyectos).
  - `belongs_to`:Grupos (Puede tener un Grupo; de hecho, es obligatorio).
  - `validates`:
    - Uniqueness: Título, Title, Abbreviated_Title, Descripcion, Description, URL_Home_Es, URL_Home_En.
    - presence:Titulo,Grupo.
  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Ocupa**
  - Indica la relación entre Despachos y Empleados que ocupan esos Despachos.
  - `belongs_to`:Empleados (Puede tener un Empleado)
  - `belongs_to`:Despachos (Puede tener un Despacho)

- **Participa_En**
  - Indica la relación entre Líneas de Investigación y Proyectos de Investigación que participan en ellas.
  - `belongs_to`:Proyectos (Puede tener un Proyecto)
  - `belongs_to`:Lineas (Puede tener un Línea)

- **Pertenence_A**
  - Indica la relación entre Titulaciones y Asignaturas de las que constan.
  - `belongs_to`:Titulaciones (Puede tener un Titulación)
  - `belongs_to`:Asignaturas (Puede tener un Asignatura)

- **Proyecto**
  - Representa la entidad Proyecto de Investigación de b.b.d.d.
  - `has_many`:Participa_En(n relaciones de participa_en con Líneas de Investigación).
  - `has_many`:Trabaja_En (n relaciones de trabaja_en con Empleados).
  - `belongs_to`:Empleados (Puede tener un Empleado Coordinador; de hecho, es obligatorio).
  - `belongs_to`:Empleados (Puede tener un Empleado Responsable; de hecho, es obligatorio).
  - `belongs_to`:Tipos_Proyecto (Puede tener un Tipo de Proyecto; de hecho, es obligatorio).
  - `validates`:
    - uniqueness: Descripcion, Description, URL_Local_Es, URL_Local_En, URL_Externo.
    - presence: Titulo, Tipo, Fecha_Inicio, Fecha_Final, Categoría, Coordinador, Responsable.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Publicación**
  - Representa la entidad Publicación de b.b.d.d.
  - `has_many`: Es_Autor (n relaciones de Es_Autor con Empleados)
  - `belongs_to`:Empleado (Puede tener un Empleado Responsable; de hecho, es obligatorio).
  - `validates`:
    - presence: Titulo, Anio, EntidadXML, Responsable.

- **Tesis**
  - Representa la entidad Tesis de b.b.d.d.
  - `belongs_to`:Empleado (Puede tener un Empleado Autor; de hecho es obligatorio).
  - `belongs_to`:Empleado (Puede tener un Empleado Director; de hecho, es obligatorio).
  - `validates`:
    - presence: Titulo, Director.
    - uniqueness: Resumen, Autor_Local, Abstract, URL_PDF.

- **Tipos_Cargo**
  - Representa la entidad Tipos de Cargo de b.b.d.d.
  - `has_many`:Empleados (n relaciones de tipos_cargo con Empleados).
  - `validates`:
    - Uniqueness: Cargo.
    - presence:Cargo.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Tipos_Contrato**
  - Representa la entidad Tipos de Contrato de b.b.d.d.
  - `has_many`:Empleados (n relaciones de tipo_contrato con Empleados).
  - `validates`:
    - Uniqueness: Contrato.
    - presence:Contrato.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Tipos_Evento**
  - Representa la entidad Tipos de Evento de b.b.d.d.
  - `has_many`:Eventos (n relaciones de tipo_evento con Eventos).

- `validates`:
  - Uniqueness: Nombre.
  - presence:Nombre.

- Métodos:
  - Valida: método que valida que la clave primaria no es nula.
  - Save_key: método que guarda la nueva clave primaria.
  - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
  - Update: método que modifica la clave primaria de una entidad ya creada.

- **Tipo_Perfil**
  - Representa la entidad Tipos de Perfil de b.b.d.d.
  - `has_many`:`Empleados(n relacionesde tipo_perfil con Empleados).`
  - `validates`:
    - Uniqueness: Perfil.
    - presence:Perfil.

  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Tipos_Proyecto**
  - Representa la entidad Tipos de Proyecto de b.b.d.d.
  - `has_many`:`Proyectos (n relaciones de tipo_proyecto con Proyectos).`
  - `validates`:
    - Uniqueness: Proyecto.
    - presence:Proyecto.

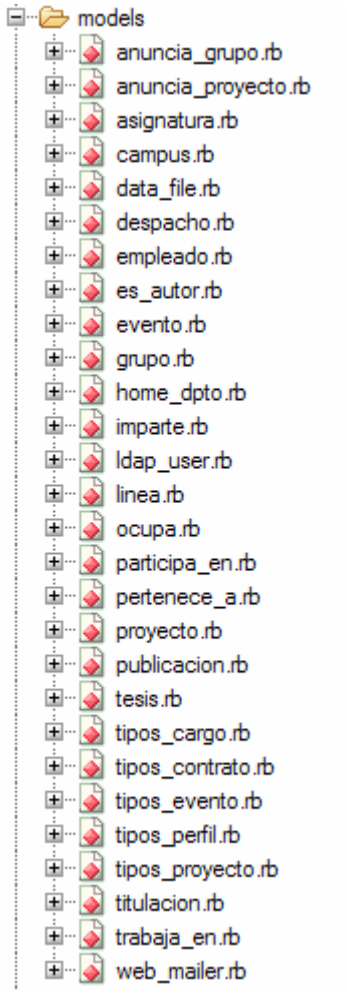
  - Métodos:
    - Valida: método que valida que la clave primaria no es nula.
    - Save_key: método que guarda la nueva clave primaria.
    - Before_create: método que guarda la clave en una variable antes de crear la entrada en b.b.d.d.
    - Update: método que modifica la clave primaria de una entidad ya creada.

- **Titulacion**
  - Representa la entidad Titulación de b.b.d.d.
  - `has_many`: `Pertenece_A (n relaciones de pertenece_a con Asignaturas).`
  - `validates`:

        - presence: Titulo, URL_Es.

- **Trabaja_En**
    - Indica la relación entre Proyectos de Investigación y Empleados que trabajan en ellos.
    - `belongs_to`:`Empleados (Puede tener un Empleado)`
    - `belongs_to`:`Proyectos (Puede tener un Proyecto)`

- **Web_Mailer**
    - Este model no representa una entidad de b.b.d.d. sino que representa las operaciones de notificaciones mediante envío de mails según la procedencia de la operación ejecutada.
    - Métodos:
        - NotificaAsignatura: método que envía un mail indicando de una operación sobre asignaturas.
        - Notificatesis: método que envía un mail indicando de una operación sobre tesis.
        - NotificaProyecto: método que envía un mail indicando de una operación sobre proyectos.
        - NotificaPublicacion: método que envía un mail indicando de una operación sobre publicaciones.
        - NotificaEmpleado: método que envía un mail indicando de una operación sobre empleados.
        - NotificaXml: método que envía un mail indicando que se ha pedido la página web personal de un empleado.


models
anuncia_grupo.rb
anuncia_proyecto.rb
asignatura.rb
campus.rb
data_file.rb
despacho.rb
empleado.rb
es_autor.rb
evento.rb
grupo.rb
home_dpto.rb
imparte.rb
ldap_user.rb
linea.rb
ocupa.rb
participa_en.rb
pertenece_a.rb
proyecto.rb
publicacion.rb
tesis.rb
tipos_cargo.rb
tipos_contrato.rb
tipos_evento.rb
tipos_perfil.rb
tipos_proyecto.rb
titulacion.rb
trabaja_en.rb
web_mailer.rb

6.1.4 Vistas.

A continuación se describirán las vistas de la aplicación, es decir, las páginas HTML de las que está compuesta. Se trata de ficheros rhtml. Básicamente son JSP adaptadas al lenguaje de Ruby, es decir, también contienen código embebido pero se trata de código Ruby.

La estructura de vistas que se ha creado está subdivida por entidades, esto es que cada entidad de la aplicación posee sus propias páginas para poder operar con el contenido de b.b.d.d. además de otras páginas que las relacionan con otras entidades o hacen más intuitivas las operaciones del usuario.

Además de las entidades, también se crearon otras vistas para la aplicación, como son menús, páginas de inicio, login, uploads, partials (trozos html y código embebido que se incluyen en otras páginas) y las páginas de contenido de mails.

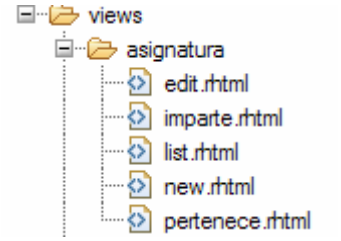
- **ASIGNATURA:**
  - Edit: Se muestra un formulario que contiene los datos actuales de la asignatura y pueden ser modificados. También se muestran dos listados: uno con el listado de profesores que imparten la asignatura y otro con las titulaciones a las que pertenece la asignatura. Ambos listados pueden ser modificados directamente en esta página (desasignar). Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - Imparte: Se muestra un listado con los empleados que NO imparten la asignatura permitiendo que cualquier número de ellos sea asignado a la asignatura.

  - List: Muestra un listado con todas las asignaturas visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil y relación con asignaturas.

  - New: Se muestra un formulario con todos los campos necesarios para crear una asignatura nueva. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

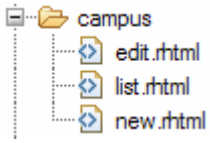
  - Pertenece: Se muestra un listado con las titulaciones a las que NO pertenece la asignatura permitiendo que ésta sea asignado a cualquier número de ellas.



- **CAMPUS:**
  - Edit: Se muestra un formulario que contiene los datos actuales del campus que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todos los campus visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un campus. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.



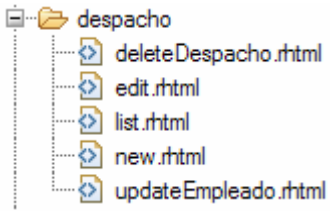
- **DESPACHO:**
  - DeleteDespacho: Se muestra un listado con los empleados que ocupan el despacho. Se les debe asignar un despacho nuevo en caso de que estos no tengan un despacho alternativo. En caso de que no se cumplan estas condiciones el despacho no será eiminado.

  - Edit: Se muestra un formulario que contiene los datos actuales del despacho que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los despachos visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil y relación del usuario con el despacho.

  - New: Se muestra un formulario con todos los campos necesarios para crear un despacho. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - UpdateEmpleado: Se muestra un listado con los despachos que no están asignados al usuario para que se le puedan asignar. Cuando se crea un empleado por defecto se accede a esta página ya que un empleado no puede crearse sin despacho. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.



- **EMPLEADO:**
  - Empleadoml: Se crea una página que contienen el cuerpo de un fichero xml con los datos personales de un empleado.

  - DeleteEmpleado: Se muestra una página donde podemos ver los puestos y acciones que desempeña el empleado seleccionado (coordinador, director, responsable, autor, etc.) Para poder eliminar al empleado deberemos reasignar un nuevo empleado de la aplicación que realice las tareas de éste. Si no se cumplen estas condiciones no se eliminará al empleado de la aplicación.

  - Despacho: Muestra un listado con todos los despachos que NO ocupa el empleado permitiendo establecer los despachos seleccionados como despachos ocupados por el empleado. Se verifica que el empleado tenga al menos un despacho.

- Edit: Se muestra un formulario con todos los campos de un empleado que son modificables. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores. También se muestra un listado con los despachos que ocupa el empleado permitiendo que sean alterados.

- List: Se muestra un listado con todos los empleados visibles al usuario filtrando por perfil y relación con el empleado permitiendo modificación y/o borrado según el mismo filtro.

- New: Se muestra un formulario con todos los campos necesarios para crear un empleado. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

- UpdateAsignatura: Se muestra la asignatura que coordina el empleado junto con un listado de empleados que pueden coordinarla para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado coordinador de la asignatura.

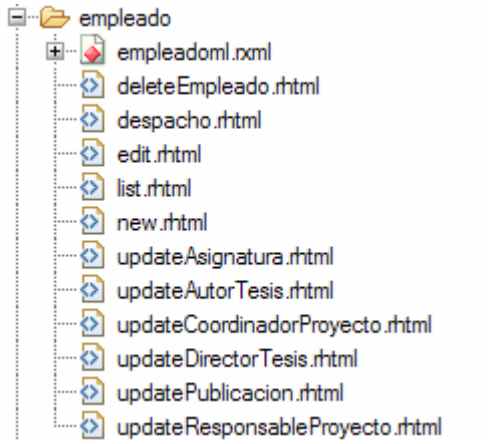
- UpdateAutorTesis: Se muestra la tesis de la que es autor el empleado junto con un listado de empleados a los que se puede asignar la autoría de la tesis para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado autor de una tesis.

- UpdateCoordinadorProyecto: Se muestra el proyecto que coordina el empleado junto con un listado de empleados a los que se puede asignar la coordinación del proyecto para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado coordinador de un proyecto de investigación.

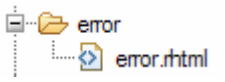
- UpdateDirectorTesis: Se muestra la tesis de la que es director el empleado junto con un listado de empleados a los que se puede asignar la dirección de la tesis para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado director de una tesis.

- UpdatePublicación: Se muestran las publicaciones de las que es responsable el empleado junto con un listado de empleados que pueden ser responsables de las publicaciónes para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado responsable de un grupo de publicaciones.

- UpdateResponsableProyecto: Se muestran los proyectos de los que es responsable el empleado junto con un listado de empleados a los que se puede asignar como responsables para que se modifique. Esto es necesario en caso de que se requiera eliminar al empleado responsable de un grupo de proyectos de investigación.



- **ERROR:**
  - Error: Se muestra una página de error con un mensaje que depende del error producido.



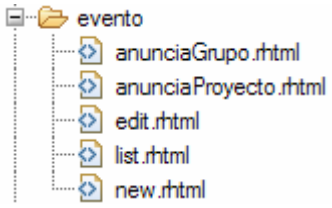
- **EVENTO:**
  - AnunciaGrupo: Se muestra un listado con los grupos en los que NO está anunciado el evento para que se seleccionen todos los que sean necesarios.

  - AnunciaProyecto: Se muestra un listado con los proyectos en los que NO está anunciado el evento para que se seleccionen todos los que sean necesarios.

  - Edit: Se muestra un formulario que contiene los datos actuales del evento que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los eventos visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un evento. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.



- **GRUPO:**
  - deleteGrupo: Para poder eliminar un grupo antes hay que reasignar grupos a todos los empleados que ahora forman el grupo, además, también habrá que reasignar grupos a las líneas de investigación creadas en el grupo a eliminar. Por ello, esta página muestra un listado con todos los empleados y las líneas de investigación del grupo permitiendo hacer estas operaciones con ellos. En caso de que no se cumplan estas condiciones no se permitirá eliminar el grupo seleccionado.

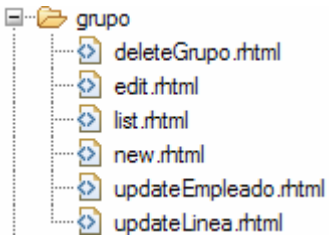
  - Edit: Se muestra un formulario que contiene los datos actuales del grupo que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los grupos visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un grupo. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - UpdateEmpleado: Se muestra una página con el empleado seleccionado y el grupo actual al que pertenece junto con un combo de grupos al que puede ser trasladado.

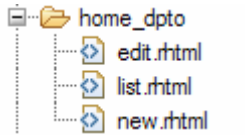
  - UpdateLinea: Se muestra una página con la línea de investigación seleccionada y el grupo al que pertence junto con un listado de grupos a los que puede ser trasladada.



- **HOME_DPTO:**
  - Edit: Se muestra un formulario que contiene los datos actuales de la "Home" del departamento que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra la Home visible al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear la Home del departamento. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

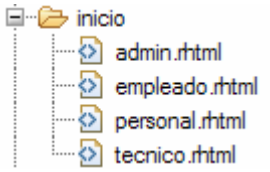


- **INICIO:**
  - Admin: Se trata del contenido de la página inicial para el perfil de ADMINISTRADOR de la aplicación. Esta puede ser modificada para que muestre cualquier noticia u operación de interés sobre la aplicación, por ejemplo, últimas modificaciones sobre usuarios o sobre otras entidades.

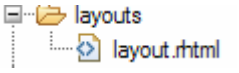
  - Empleado: Se trata del contenido de la página inicial para el perfil de EMPLEADO de la aplicación. Puede ser modificada para que muestre noticias importantes para el usuario, por ejemplo, cambios en sus datos o eventos nuevos.

  - Personal: Se trata del contenido de la página inicial para el perfil de ADMINISTRATIVO de la aplicación. Esta puede ser modificada para que muestre cualquier noticia u operación de interés sobre la aplicación, por ejemplo, últimas modificaciones sobre usuarios o sobre otras entidades controladas por ellos.

  - Tecnico: Se trata del contenido de la página inicial para el perfil de TECNICO de la aplicación. Puede ser modificada para que muestre noticias importantes para el usuario, por ejemplo, usuarios nuevos que cuyos datos deben ser modificados.



- **LAYOUTS:**
  - Layout: Se trata del contenido del menú lateral izquierdo para el usuario. Este menú es constante y muestra las operaciones permitidas en él dependen del perfil del usuario logado.

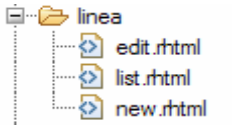


- **LINEAS:**
  - Edit: Se muestra un formulario que contiene los datos actuales de la línea de investigación que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

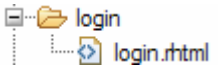
  - List: Muestra un listado con todas las líneas visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil (está visión ya no está disponible según el

workflow de la aplicación debido a modificaciones durante el proyecto pero aún podría ser accedida de manera artificial componiendo una URL).

- New: Se muestra un formulario con todos los campos necesarios para crear una línea de investigación para un grupo. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores. Este formulario es obligatorio para todas las líneas salvo las líneas por defecto de un grupo que son creadas de forma transparente al usuario de la aplicación cuando éste crea un grupo.



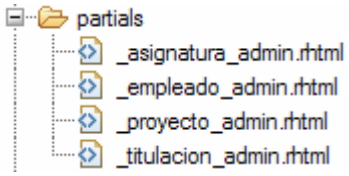
- **LOGIN:**
  - Login: Se trata de la página inicial de la aplicación. Contiene un formulario que pide al usuario su login y password para poder logarse. En caso de error de login, se redirige la página al mismo lugar.



- **PARTIALS:**

Se trata de una pequeña porción de código ruby embebido en una página rhtml que será incluida en otra página de la aplicación atendiendo al perfil del usuario. El nombre de los partials siempre debe comenzar por "_".

- asignatura_admin: En este caso permite crear una nueva asignatura si el perfil lo permite.
- empleado_admin: En este caso permite crear un nuevo empleado si el perfil lo permite.
- proyecto_admin: En este caso permite crear un nuevo proyecto si el perfil lo permite.
- titulacion_admin: En este caso permite crear una nueva titulación si el perfil lo permite



- **PROYECTO:**
  - Edit: Se muestra un formulario que contiene los datos actuales del proyecto que pueden ser modificados. También se muestra un listado con todos los empleados que trabajan en este proyecto permitiendo eliminarlos o añadir nuevos empleados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

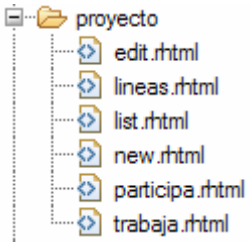
- Lineas: Se muestra un listado con las líneas de investigación en las que participa el proyecto actualmente permitiendo que se eliminen o se añadan nuevas líneas. Se puede añadir la línea por defecto de un grupo pero si se añade otra línea del mismo grupo, la relación entre el proyecto y la línea por defecto no se mostrará al usuario (petición expresa durante el desarrollo del proyecto: es la solución encontrada para añadir un proyecto a un grupo entero en el caso en que no se distingan diferentes líneas de investigación para un grupo).

- List: Muestra un listado con todas los proyectos visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil y según la relación del usuario con el proyecto.

- New: Se muestra un formulario con todos los campos necesarios para crear un proyecto. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

- Participa: Se muestra un listado con todas las líneas de investigación de un grupo anteriormente seleccionado disponibles y en las que el proyecto NO participa permitiendo que el usuario agregue el proyecto a estas líneas.

- Trabaja: Se muestra un listado con todos los empleados que NO trabajan en un proyecto permitiendo al usuario que los incluya en el proyecto. Tanto cuando se crea como cuando se modifica un proyecto se verifica que éste cuenta con al menos un empleado trabajando en él. En caso contrario se fuerza al usuario a introducir un empleado que trabaje en él.



- **PUBLICACION:**
  - Autor: Se muestra un listado con los empleados que NO son autores de la publicación seleccionada permitiendo que estos sean modificados o añadidos como autores de esta. En cualquier caso se fuerza al usuario a que establezca un autor para todas las publicaciones creadas aunque tenga que inscribirse temporalmente a él mismo como autor (petición expresa durante el desarrollo del proyecto) sin embargo esta autoría puede ser modificada en cualquier momento.

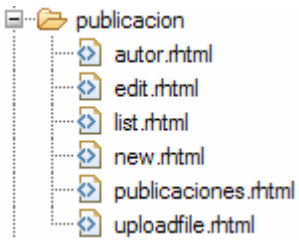
  - Edit: Se muestra un formulario que contiene los datos actuales de la publicación que pueden ser modificados. También se muestra un listado con todos los empleados que son autores de ella permitiendo eliminarlos o añadir nuevos autores. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

- List: Muestra un listado con todas las publicaciones visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil y según la relación del usuario con el proyecto (autor, responsable).

- New: Se muestra un formulario con todos los campos necesarios para crear una publicación. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores (por petición expresa durante el desarrollo del proyecto se ha eliminado esta opción del workflow de la aplicación siendo sustituida por uploadfile).

- Publicaciones: Se muestra un listado con todas las publicaciones creadas durante la carga del fichero. Se establece esta página para que el usuario que ha cargado dichas publicaciones pueda establecer los autores de estas. Se obliga al usuario a que permanezca en esta página hasta que establezca los autores para las publicaciones creadas.

- Uploadfile: Se muestra un formulario para subir un fichero XML con las publicaciones que el usuario quiera crear. Si el formulario no sigue el estándar establecido o se produce algún error, se comunica al usuario y se retorna a la pantalla inicial.



- **TESIS:**
  - EditDoctorado: Se muestra un formulario que contiene los datos actuales de la tesis doctoral que pueden ser modificados. Debido a que se trata de un doctorado se muestran sólo los datos correspondientes a este tipo de tesis (Autor).

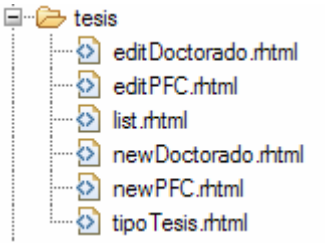
  - EditPFC: Se muestra un formulario que contiene los datos actuales de la tesis de proyecto de fin de carrera que pueden ser modificados. Debido a que se trata de un PFC se muestran sólo los datos correspondientes a este tipo de tesis.

  - List: Muestra un listado con todas las tesis visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil y según la relación del usuario con las tesis (se mostrarán PFC y Doctorados indistintamente).

  - NewDoctorado: Se muestra un formulario con todos los campos necesarios para crear un Doctorado. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

- NewPFC: Se muestra un formulario con todos los campos necesarios para crear un PFC. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

- TipoTesis: Se muestra al usuario una selección de Doctorado o PFC para proceder a crear ese tipo de tesis.



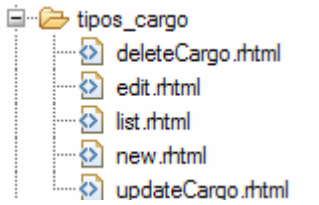
- **TIPOS_CARGO:**
  - DeleteCargo: Se muestra un listado con los empleados que poseen el cargo seleccionado. Se les debe asignar un cargo nuevo antes de poder borrar el actual. En caso de que no se cumplan estas condiciones el tipo de cargo no será eliminado.

  - Edit: Se muestra un formulario que contiene los datos actuales del tipo de cargo que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los tipos de cargo visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un tipo de cargo. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - UpdateCargo: Se muestra un listado con los cargos que se pueden asignar al empleado. Sólo se accede a esta página en caso de que se quiera eliminar un tipo de cargo.



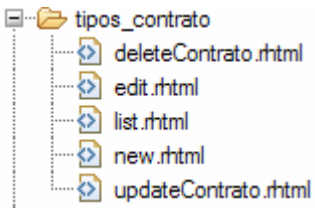
- **TIPOS_CONTRATO:**
  - DeleteContrato: Se muestra un listado con los empleados que poseen el contrato seleccionado. Se les debe asignar un contrato nuevo antes de poder borrar el actual. En caso de que no se cumplan estas condiciones el tipo de contrato no será eliminado.

  - Edit: Se muestra un formulario que contiene los datos actuales del tipo de contrato que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los tipos de contrato visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un tipo de contrato. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

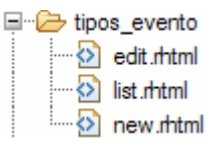
  - UpdateContrato: Se muestra un listado con los contratos que se pueden asignar al empleado. Sólo se accede a esta página en caso de que se quiera eliminar un tipo de contrato.



- **TIPOS_EVENTO:**
  - Edit: Se muestra un formulario que contiene los datos actuales del tipo de evento que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todos los tipos de evento visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un tipo de evento. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.



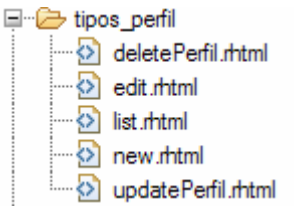
- **TIPOS_PERFIL:**
  - DeletePerfil: Se muestra un listado con los empleados que poseen el perfil seleccionado. Se les debe asignar un perfil nuevo antes de poder borrar el actual. En caso de que no se cumplan estas condiciones el perfil no será eliminado.

  - Edit: Se muestra un formulario que contiene los datos actuales del tipo de perfil que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los tipos de perfil visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un tipo de perfil. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - UpdatePerfil: Se muestra un listado con los perfiles que se pueden asignar al empleado. Sólo se accede a esta página en caso de que se quiera eliminar un tipo de perfil.



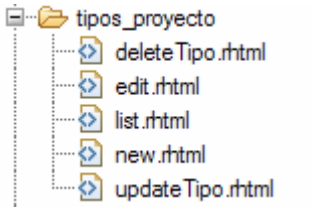
- **TIPOS_PROYECTO:**
  - DeleteTipo: Se muestra un listado con los proyectos que poseen el tipo seleccionado. Se les debe asignar un tipo nuevo antes de poder borrar el actual. En caso de que no se cumplan estas condiciones el tipo de proyecto no será eliminado.

  - Edit: Se muestra un formulario que contiene los datos actuales del tipo de proyecto que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - List: Muestra un listado con todas los tipos de proyecto visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear un tipo de proyecto. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - UpdateTipo: Se muestra un listado con los tipos de proyecto que se pueden asignarl a un proyecto. Sólo se accede a esta página en caso de que se quiera eliminar un tipo de proyecto.



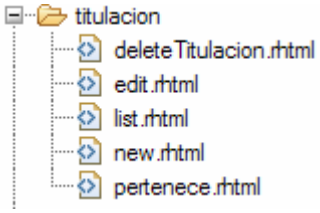
- **TITULACION:**
  - DeleteTitulacion: Se muestra un listado con los las asignaturas que se imparten en la titulación seleccionada. Se les debe asignar una titulación diferente antes de poder borrar la actual. En caso de que no se cumplan estas condiciones la titulación no será eliminada.

  - Edit: Se muestra un formulario que contiene los datos actuales de la titulación que pueden ser modificados. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores. Se muestra también un listado con las asignaturas que se imparten en la titulación permitiendo que sean modificadas.

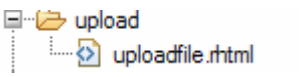
  - List: Muestra un listado con todas las titulaciones de la aplicación visibles al perfil y al usuario actual. Se le permitirá modificar y/o borrar según perfil.

  - New: Se muestra un formulario con todos los campos necesarios para crear una nueva tutulación. Si el usuario introduce erróneamente datos o deja de introducir datos necesarios, la aplicación redirige a esta misma página indicando los errores.

  - Pertenece: Se muestra un listado con las asignaturas disponibles en la aplicación que NO pertenecen a la titulación para que estas puedan serlo.



- **UPLOAD:**
  - uploadfile: Página para poder subir cualquier tipo de archivo a la aplicación (por petición durante el desarrollo se eliminó esta idea y sólo se permitirá la subida de ficheros XML con el formato Bibtexml)



- **WEB_MAILER:**
  - NotificaXml: Se crea el cuerpo de un fichero xml con los datos personales de un empleado.

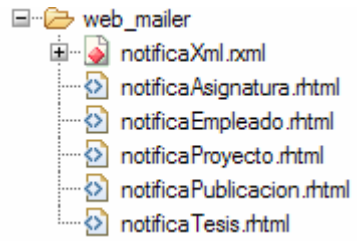
  - NotificaAsignatura: Necesita ser creada para el mantener el workflow de envío de mails debido a operaciones realizadas sobre asignaturas.

  - NotificaEmpleado: Necesita ser creada para el mantener el workflow de envío de mails debido a operaciones realizadas sobre empleados.

  - NotificaProyecto: Necesita ser creada para el mantener el workflow de envío de mails debido a operaciones realizadas sobre proyectos.

  - NotificaPublicacion: Necesita ser creada para el mantener el workflow de envío de mails debido a operaciones realizadas sobre publicaciones.

  - NotificaTesis: Necesita ser creada para el mantener el workflow de envío de mails debido a operaciones realizadas sobre tesis.



6.2. Directorio de Configuración.

Aquí se encuentran los ficheros de configuración de la aplicación.

Se compone de varios ficheros donde se describe la configuración del servidor que lanza la aplicación así como de otros ficheros que describen la conexión a b.b.d.d o servidores externos.

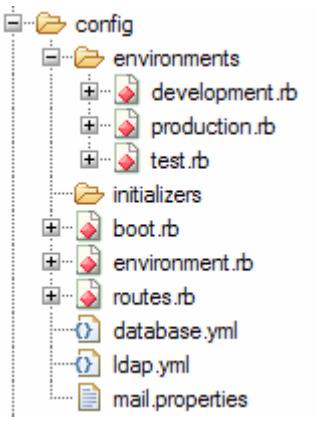
Los más importantes:

- Environment.rb: En este fichero decidiremos en que entorno (desarrollo, pruebas o producción) se lanza la aplicación. También se decide el tipo de servidor que corre la aplicación.

- Database.yml: En este fichero se describe la configuración de la conexión a base de datos de la aplicación. Podremos configurar los tres entornos descritos anteriormente.

- Ldap.yml: En este fichero se describe la configuración de la conexión al servidor LDAP de la aplicación.

- Mail.properties: En este fichero se describe la configración del servidor de correo de la aplicación.



7. Desarrollo y Trabajos Futuros.

A lo largo del desarrollo de la aplicación nos hemos encontrado numerosos momentos en los que se ha tenido que considerar el diseño inicial de la aplicación y modificarlo para poder continuar en el desarrollo.

Esto era debido a que el contenido de las páginas Web del departamento de telemática no mantiene un formato común sino que cada grupo (GAST, NETCOM) pueden diseñar estas páginas como se le antojen con el consiguiente problema para abstraer la estructura de la página y los elementos que la forman. Igualmente, ambos grupos no se comportan de igual manera a la hora de relacionarse con proyectos u otras entidades definidas para la aplicación por lo que las varibles a considerar a la hora de mostrar/guardar la información dependen del grupo del usuario.

Además, el principal escollo que nos encontramos a la hora de realizar este proyecto fue la coordinación entre dos estudiantes para crear un punto de unión de ambas aplicaciones, en este caso fue la Base de Datos. De esta forma hubo que exponer ideas hasta llegar a un modelo más o menos estable con el que empezar el desarrollo de los proyectos. Este modelo, por supuesto, fue variando progresivamente atendiendo a las demandas de los proyectandos y de la propia información a mantener.

A continuación se enumerarán los problemas que encontramos en el proyecto y si es posible una solución sencilla para ellos.

7.1 Problemas de Diseño de Base de Datos.

Observando la interacción entre la aplicación y la b.b.d.d. encontré que la mayoría de las tablas tienen un error de diseño. Se utiliza como clave primaria de las tablas campos que posteriormente pueden ser modificados por los usuarios. Esto no es recomendable para nuestra aplicación ya que contando con la interacción simultánea de usuarios se puede dar, por ejemplo, el caso de que un ADMINISTRADOR o ADMINISTRATIVO este modificando uno de estos valores de b.b.d.d. de una entidad concreta cuando un EMPLEADO esté accediendo a esa misma entidad. En este caso, la operación del EMPLEADO siempre dará error ya que sus operaciones tienen como referencia la clave primaria inicial que ahora es diferente al haberla modificado uno de los perfiles administrativos. Este caso choca con los mapeos de Objetos de BBDD de Ruby on Rails por lo que es necesario sobreescribir el método de modificación de un objeto para poder realizar la actualización en caso de que se intente modificar el campo que actúa como clave primaria.

La forma de solucionar sería replantear el diseño de base de datos para que se incluya un campo ID Integer autoincrementado (ya que estamos usando MySQL) para cada entidad, en caso de que no cuenten ya con él. Este ID sería no visible a las operaciones del usuario por lo que no sería modificable.

Incoherencias entre UML y modelo de Base de Datos:

Para poder comprender mejor la relación entre tablas de b.b.d.d (multiplicidad, comportamiento, etc) y aplicar esto al comportamiento del código, también se hizo necesario estudiar el modelo UML que realizó el proyectando **D. Carlos Holgado Molinillo** y que viene detallado en el anéxo A de esta memoria. Durante este estudio se detectaron posibles incoherencias entre el modelo UML y el de Base de Datos que se enumeran acontinuación:

- Empleados: A la hora de indicar el Grupo de un Empleado, se define que según el cargo que éste desempeña podrá tener o no un grupo asignado. El modelo descrito no concuerda con el modelo real (descrito en los templates de casos de uso descritos en el capítulo 10 de esta memoria).

- Cargos: Se indica en el modelo UML que todo Empleado tiene que tener un cargo asignado pero en el modelo de b.b.d.d. se indica que este campo puede ser NULL por lo que ningún cargo será asignado al Empleado.

- Tesis: Existe una incoherencia en el modelo a la hora de definir el "Autor Local" de una tesis ya que este campo cambia cuando se trata de un PFC o un Doctorado.

Todas estas incoherencias han sido tratadas con el director del proyecto para que en la implementación se siga el comportamiento deseado inicialmente.

Convendria revisar, y modificar si fuera necesario, el diseño de la b.b.d.d en iteraciones posteriores para verificar la completa veracidad entre ambos modelos en caso de que en esta primera vista general no se haya encontrado más incoherencias de este tipo.

7.2 Problemas de Autenticación de Usuarios.

Debido a que esta aplicación fue pensada inicialmente para un tipo de usuario (ADMINISTRADOR), podemos encontrar que al incluir nuevos perfiles a la aplicación las acciones que estos realizan no controlan completamente la identidad del usuario.

Los Controllers de una Entidad cuentan con métodos de autenticación de perfil, es decir, usuarios que no tengan el perfil necesario no podrán acceder a ninguno de los métodos de los Controllers si no tienen derechos de operación sobre esa Entidad. Por ejemplo un EMPLEADO nunca podrá acceder a ninguna de las operaciones sobre Campus ya que su perfil no tiene los permisos necesarios. Sin embargo si el usuario tiene permisos de alguna clase sobre una entidad, el Controller permitirá al usuario la visión de los métodos sobre esa entidad.

Se controla las acciones de los empleados según la Vista. Por ejemplo, a un EMPLEADO se le permite visualizar un listado con todas las asignaturas que controla y por ello tendrá permisos de modificación sobre esta asignatura. No se controla si el EMPLEADO que modifica la asignatura es su coordinador ya que en principio sólo se le han mostrado las asignaturas que coordina.

De esta forma, si algún usuario de la aplicación conociese las rutas de los métodos de los controladores y los parámetros que estos utilizan podría hacer un uso malintencionado de estos datos (generando url's artificiales) y realizar operaciones sobre entidades que no le correspondan directamente. Es decir, podría asumir la identidad de otro EMPLEADO con los mismos permisos.

Existen varias formas de controlar esto:

- Se podrían modificar los métodos de actualización de entidades para que antes de realizar ninguna operación sobre b.b.d.d. se verifique que la propiedad de la entidad con la que se va a operar se corresponde con la entidad del usuario que está operando o si éste tiene los permisos necesarios.
- Se podría modificar el diseño de b.b.d.d. y añadir historiales de modificación, es decir, guardar en b.b.d.d. el usuario que modifico la entidad y en qué momento se hizo. De esta forma, siempre contando con que se dispone de un back-up de b.b.d.d. se podría volver al estado anterior a la modificación en esa entidad.

7.3 Problemas de Operaciones con Líneas de Investigación.

Cada grupo de la aplicación dispone de una línea de investigación creada por defecto con los mismos parámetros con los que se creó el grupo. La única forma de distinguir si una línea de investigación es la línea por defecto de un grupo es verificando si esa línea se corresponde en su clave primaria con alguno de los grupos que existen.

Durante el desarrollo he podido contemplar que esto provoca un posible error cuando trabajando con proyectos, se quiere asignar un proyecto a una línea de investigación.

Sucede que al no contar con una distinción clara (flag en b.b.d.d.) para qué líneas son por defecto y cuáles no, el listado de posibles líneas a las que asignar el proyecto puede aparecer incompleto ya que se quieren mostrar todas las líneas en las que el proyecto aún no está asignado pero también se quiere evitar mostrar aquellas que son líneas por defecto de un grupo que tiene otras líneas a las que el proyecto está asignado. No se puede crear una query que refleje esto directamente.

Una forma de solucionar esto es modificar la b.b.d.d. para que se incluya un campo nuevo en las líneas de investigación que refleje si son líneas por defecto y si lo son a qué grupo pertenecen, por ejemplo un campo que indique la relación con grupos (IDGRUPO en caso de que sea una línea por defecto o NULL en caso de que no lo sea).

7.4 Problemas con el lenguaje de programación.

Inicialmente Ruby parecía ser una buena opción de lenguaje para el desarrollo del proyecto pero según avanzamos en el desarrollo encontramos cada vez más problemas con este lenguaje.

Quizás por su temprana edad, podemos observar que Ruby aún tiene muchas funcionalidades a crear y otras muchas a mejorar.

La documentación que se puede encontrar de cada librería es muy escueta sino nula. Además, cada vez que se crea una nueva versión de la librería se alteran dependencias con otras librerías lo que generalmente provoca que nuestros programas no funcionen.

Las excepciones que lanza Ruby no son fáciles de comprender y en ocasiones el mensaje mostrado no se correspondería con el verdadero error ocurrido.

Otra situación que se ha dado durante el desarrollo del proyecto es que Ruby distingue entre mayúsculas y minúsculas cuando se refiere a nombres de tablas o columnas en b.b.d.d. lo que produjo numerosos fallos cuando se trató de coordinar los dos proyectos de los que consta el proyecto de WEB DEPARTAMENTAL cuando se modificaba el nombre de una columna de una tabla. Se producía entonces que la clase que hacía referencia a esta tabla provocaba un error de compilación al no poder mapear de forma sencilla la columna anterior a la variable actual por lo que había que reescribir el nombre de este campo de la entidad en todas aquellas clases en donde apareciese.

Como solución a estos problemas encontrados en las librerías de Ruby, plantearía un rediseño del lenguaje de programación y aplicaría JRuby que se trata de una implementación de Ruby 100% java por lo que se podría usar sin ningún problema librerías de este lenguaje. Este cambio no plantearía más problema que cambiar la versión de Ruby utilizada hasta ahora por la nueva versión de JRuby on Rails.

También se pueden producir errores con este lenguaje cuando se modifica la versión de ruby a ejecutar. Se da el caso que que si quisiéramos trasladar un proyecto completo funcionando con una versión de Ruby a otro servidor con otra versión (superior) de Ruby instalada, se pueden provocar errores de compilación al haber desaparecido métodos de una versión a otra superior. Este sucedió cuando se trató de crear el prototipo de pruebas de la aplicación en los servidores de la universidad. Se puede solucionar este error bien modificando los scripts que lanzan el servidor (WEBrick) bien creando un proyecto con igual nombre en el servidor destino (se creará un esqueleto de aplicación común) e ir copiando sólo las clases de la carpeta "app" y "Config" de origen una a una y según convenga (algunos ficheros de la carpeta de configuración no deberían ser modificados, por ejemplo boot.rb) pero este hecho hace que los proyectos Ruby no sean cómodos de trasladar tal y como lo haríamos con un proyecto escrito en otro lenguaje, por ejemplo Java.

Uno de los principales problemas encontrados a la hora de programar con ruby on rails ha sido la versión de trabajo. Durante el desarrollo de la aplicación se utilizó generalmente la versión de rails 2.0. Cuando finalmente el proyecto fue acabado y se pasó a migrar al servidor de la universidad, surgió el problema de versiones entre diferentes S.O. En este

caso se programabe en Windows XP (rails 2.0) mientras que el servidor de producción sería un Debian (rails 1.1.6). Esta diferencia entre versiones, y el problema que surgía de la actualización de versión en Debian (a rails 2.1) provocaba un nuevo problema: las aplicaciones desarrolladas en rails 2.0 normalmente tienen incompatibilidades con la versión 2.1 que solamente pueden ser arreglados con plugins y fragmentos de código para solucionar dependencias en las librerías.

Como solución a este problema, rails cuenta con el directorio "vendor" en el que podemos copiar todas las librerías de proyectos rails 2.0 sin tener en cuenta la versión de rails que este funcionando en el servidor y habrá que modificar sólo algunos parámetros de configuración en el fichero "environment.rb" de la aplicación para que se tenga en cuenta este directorio. Con esta solución se salva el problema de la versión instalada pero habrá que copiar a este directorio todas las librerías que se necesiten junto con sus dependencias por lo que este proceso también puede llegar a complicarse.

7.5 Problemas de Login mediante LDAP.

Este sería un problema heredado del anterior (Problemas con el lenguaje elegido). Inicialmente se desarrollo un login LDAP con las librerías de active-ldap de Ruby que permitían una conexión con cualquier servidor LDAP para poder realizar el login de la aplicación.

Al modificarse la versión de Ruby (upgrade), se detectó que la librería de active-ldap había sufrido grandes modificaciones y se habían omitido algunos de los métodos de conexión de la librería anterior, es decir, se producían errores de compilación y por supuesto de conexión.

Finalemente se pudo desarrollar un sistema de login mediante ldap gracias a la librería net-ldap de Ruby (desarrollada en Ruby puro) pero esto produjo grandes cambios y largas pruebas hasta desarrollar un sistema de login nuevo a partir de una librería diferente.

Como solución plantearía nuevamente el uso de JRuby on Rails para el desarrollo de la aplicación, al menos mientras que las actualizaciones de Ruby no impliquen cambios tan drásticos en sus librerías.

7.6 Formato de XML de carga de publicaciones.

En principio se ideó este sistema para hacer cargas masivas de publicaciones de un empleado, bien por él mismo bien por un administrador o un administrativo de la aplicación.

Se ha establecido que el formato del XML de entrada será BibTeXML (en implementaciones futuras se permitirán formatos diferentes a este según se contempló en el diseño del proyecto).

Este formato aun tratándose de un estándar, tiene varias implementaciones por lo que esto podría provocar conflictos a la hora de presentar los datos según el proyecto de "Análisis, diseño e implementación de un sitio Web Departamental: almacenamiento, recuperación y presentación de contenidos" desarrollado por **D. Carlos Holgado Molinillo**.

Esto es debido a que para la aplicación que contemplamos ahora, es transparente el prefijo de las etiquetas xml del documento (impuestas por el creador del documento) por lo que cuando un usuario cree un documento BibTeXML y lo suba a la aplicación, ésta lo formateará siguiendo el dtd específico para BibTeXML pero sin tener en cuenta los prefijos(los añadirá tal cuál en b.b.d.d.). Habría que hacer notar que el proyecto encargado de presentarlos datos de las publicaciones es dependiente de estos prefijos, tiene un prefijo establecido, por lo que sería recomendable modificar esto para hacerlo independiente.

Habría que asegurar que el otro proyecto que complementa a este tenga esto en cuenta (los documentos xslt deberían filtrar por nombre sencillo de etiqueta XML (es decir, sin prefijo) y no por nombre cualificado de etiqueta XML (es decir, con prefijo) o de lo contrario se producirán errores en la presentación en caso de que el usuario que ha introducido las publicaciones no lo haga con un prefijo soportado por los xslt).

7.7 Definición de las acciones de cada perfil.

Las acciones que les están permitidas a los usuarios según sus perfiles no están totalmente definidas. Aún estarían por delimitar claramente los perfiles de ADMINISTRATIVO y TÉCNICO. Esto desembocará en un ligero rediseño de la aplicación para delimitar claramente qué acciones pueden realizar estos dos perfiles.

Actualmente los perfiles ADMINISTRADOR y ADMINISTRATIVO tienen prácticamente los mismos poderes dentro de la aplicación salvo a la hora de modificar los datos de los Empleados, en cuyo caso los ADMINISTRATIVOS tienen vedados algunos de los campos. Según este sistema, un usuario ADMINISTRATIVO podría cancelar o modificar acciones realizadas por un usuario ADMINISTRATIVO por lo que ambos perfiles se equipararían. Incluso se podría dar el caso de que un usuario ADMINISTRATIVO eliminase a un usuario ADMINISTRADOR.

Ya que el sistema de perfiles se estableció como una pirámide en la que el perfil ADMINISTRADOR tiene una visión absoluta de la aplicación, habría que limitar, al menos, la visión que tienen el resto de perfiles, es decir, un usuario ADMINISTRATIVO no debería poder ver usuarios ADMINISTRADORES (de esta forma evitaríamos que un usuario ADMINISTRATIVO pudiese eliminar a un ADMINISTRADOR accidentalmente). Además vuelvo a recomendar mantener un histórico de operaciones sobre los objetos para que en caso de que un ADMINISTRATIVO modifique una operación realizada por un ADMINISTRADOR está operación pueda ser localizada y rescatada.

Al perfil de TÉCNICO sólo se le han dado permisos para modificar los campos de usuario y login de un empleado. No se ha profundizado sobre este perfil pero quizá se le podrían otorgar mayores permisos de modificación sobre un empleado.

Otro punto que sería interesante de desarrollar en una futura iteración de la aplicación sería la posibilidad de que los usuarios poseyeran multiples perfiles para poder desempeñar varias funcionalidades.

7.7.1 Sistema multi-perfil y multi-permiso.

1. **Sistema multi-perfil:**

Se ha estudiado para las siguientes iteraciones de la aplicación el proporcionar a los usuarios varios perfiles y que sean ellos los que decidan entrar en la aplicación con unos u otros perfiles. Esto se pensó por los casos de los perfiles especiales (ADMINISTRADOR, ADMINISTRATIVO y TÉCNICO) ya que estos usuarios pueden llegar a ser también empleados de un departamento en concreto y quizá les sea útil acceder a la aplicación con el perfil EMPLEADO de ese departamento. De esta forma, sería la aplicación la que preguntaría al usuario que perfil quiere usar para navergar por ella.

2. **Sistema multi-permiso:**

Pensado en posibles interacciones entre usuarios o la posibilidad de casos especiales entre ellos (vacaciones, enfermedad, etc), se ha ideado un sistema binario para dar permisos sobre la aplicación al usuario que lo necesite independientemente del perfil que tenga. Así, no es necesario modificar el perfil de ningún usuario sino que basta con darle permisos sobre las acciones que necesite.

De manera similar a la categoría de los eventos, se destinaría un campo de b.b.d.d. de EMPLEADOS para indicar los permisos que el usuario que está accediendo a la aplicación tiene. Esto es, se indica una cadena de 0 y 1 por los que según la posición del carácter y su valor (0/1) podemos llegar a si un usuario tiene unos permisos determinados sobre una determinada acción dentro de la aplicación (similar a los permisos sobre ficheros de UNIX).

7.8 Otros trabajos futuros.

- **Servidor http:**

Debido a que este proyecto está todavía en fase de pruebas, es recomendable continuar usando WEBrick hasta que se decida aplicarlo en un entorno de producción ya que este servidor se crea cuando lanzamos el script del esqueleto de la aplicación en ruby ($ ruby "nombre_aplicación") y además sus parámetros son muy sencillos de configurar (tanto los parámetros de puerto y dirección en los que desplegarse como los parámetros del entorno en el que corre: environment.rb --- con sus respectivos ficheros de configuración development, test y production).

En el anéxo C de este texto se indica como instalar la aplicación y lanzarla con WEBrick como servidor HTTP. También se muestran los pasos para lanzarla bajo un servidor Apache (indendiente del servidor de datos que podría ser WEBrick o Mongrel).

La recomendación más común en los entornos de producción en los que se opera con Ruby on Rails es usar un servidor Apache como balanceador de carga y varios servidores Mongrel como servidores de datos.

8. Conclusiones.

En este punto se evaluará si la aplicación desarrollada cumple con los objetivos definidos en el punto 4 de este texto.

8.1 Interfaz Web con datos del Departamento.

La aplicación que nos encontramos al finalizar el desarrollo del proyecto se trata de una herramienta sencilla para el control del contenido de las páginas web de un departamento de la universidad, en concreto del departamento de Telemática, aunque su planteamiento permitirá que pueda ser aplicado a otros departamentos de la universidad que sigan un modelo parecido.

Esta herramienta permite el acceso, creación y modificación de cualquier dato dentro de la b.b.d.d. del departamento mediante formularios web. Además, ha sido diseñada para que las operaciones sobre los datos sigan la política del departamento.

8.2 Sistema de control de acceso a la información.

Se han establecido dentro de la aplicación cuatro perfiles (ADMINISTRADOR, ADMINISTRATIVO, EMPLEADO y TÉCNICO).

Según el diseño de la aplicación y la lógica implementada, se ha dotado a cada perfil de unas funciones específicas (salvo en el caso especial de ADMINISTRADOR y ADMINISTRATIVO donde ambos perfiles comparten prácticamente todas las funcionalidades) de forma que se establecen permisos graduales. También se han creado funcionalidades específicas para cada perfil (casos como Web Personal para EMPLEADO o control de Login local para los TÉCNICOS).

El diseño que se ha seguido pretende que la interacción entre el usuario y la aplicación sea lo más sencilla posible tratando en todo caso de limitar los errores al usuario, es decir, que no se produzcan errores inesperados por acciones de los usuarios o por lla vista que tienen estos sobre las entidades de la base de datos (salvo por los puntos de seguridad sobre usuarios discutidos en el capítulo 7 de este texto).

Todos los usuarios que se hagan login en la aplicación dispondrán de un menú sencillo que les indicará en todo momento que acciones les está permitadas realizar. También se controlan las acciones disponibles en todos los listados de las entidades a las que tienen acceso, por ejemplo, un usuario EMPLEADO sólo podrá realizar acciones sobre sí mismo (modificar), pero además se le prohíbe que borre su usuario o que modifique datos que se salgan de los puramente personales.

8.3 Integración con Presentación de Datos.

Esta aplicación se trata de una pieza importante a acoplar con el proyecto de "Análisis, diseño e implementación de un sitio Web Departamental: almacenamiento, recuperación y publicación de contenidos" realizado por el proyectando D.Carlos Holgado Molinillo. Ambos proyectos se complementan, uno para controlar y almacenar el contenido de las páginas web y otro para recuperarlos y presentarlos.

Estos dos proyectos requerirán de un periodo de pruebas conjuntas hasta que se verifique el correcto funcionamiento entre ambos. Además, durante este periodo se comprobará también si el diseño de ambos proyectos es suficiente para contener y mostrar toda la información que se desea en la páginas del departamento o si alguno de los dos (o ambos) necesitan de una reestructuración de su diseño.

Es aquí donde puntualizo que se necesitará una iteración sobre la aplicación para resolver, al menos, los problemas localizados en el punto 7 (Desarrollo) de este texto. Ya que estos comprometen el futuro uso de esta aplicación como herramienta estable dentro del departamento de telemática para presentar el contenido de sus páginas web.

Estas aplicaciones tienen como único punto de unión la Base de Datos del departamento por lo que en caso de que se produzca una iteración sobre alguna de las dos versiones, lo único que se necesita es verificar que se opera de igual manera en anbas versiones con los datos de la Base de Datos, es decir, ambas aplicaciones son modulares y pueden ser versionadas de forma independiente sin afectar al funcionamiento de la otra.

8.4 Sistema multiplataforma.

La aplicación ha sido totalmente implementada y probada sobre un entorno de desarrollo (DEV) en el Sistema Operativo Windows XP. Aquí se hicieron sucesivas pruebas durante la implementación de la aplicación realizadas por el proyectando D. Adolfo Miguel Catalán García-Manso y verificadas durante las sesiones de seguimiento del proyecto por los directores de este proyecto.

Posteriormente, se ha trasladado este código a un servidor Devian de la universidad (trompa.it.uc3m.es) creando de esta manera un prototipo de pruebas para usuarios (entorno UAT). En este entorno también se han realizado pruebas realizadas por los usuarios mencionados anteriormente. También se han realizado pruebas de integración con el proyecto de "Análisis, diseño e implementación de un sitio Web Departamental: almacenamiento, recuperación y publicación de contenidos" realizado por el proyectando D.Carlos Holgado Molinillo.

Según las pruebas realizadas en ambos Sistemas operactivos, podemos concluir que la aplicación que ha sido desarrollada se trata de un sistema multiplataforma.

8.5 Sistema multiidioma.

La propia implementación de la b.b.d.d. garantiza que la información resultante pueda ser mostrada tanto en Castellano como en Inglés (salvo por pequeños datos). En principio no ha sido necesaria la implementación de esta aplicación para ser presentada al usuario en diferentes idiomas (principalmente porque se realiza para el departamento de telemática de la universidad) por lo que se considera "multiidioma" a la capacidad de la aplicación de guardar información en varios idiomas y poder acceder a esta información de forma similar.

8.6 Sistema flexible.

Podemos decir también que el sistema creado es muy flexible y que con ligeras modificaciones de código podemos obtener una herramienta muy potente para:

- Generar Web de otros grupos: Esto está ya completamente funcional en esta versión de la aplicación. Simplemente habría que añadir la información de estos grupos a la b.b.d.d. y lanzar la aplicación que genera estas páginas.

- Web de Líneas de investigación: Sucede igual que con los grupos, simplemente hay que incluir esta información en b.b.d.d. y generar las páginas.

- Cálculo de estadísticas:
  - Informes para el departamento (publicaciones, proyectos, etc).
  - Listas de correo por grupo, proyecto, departamento, etc.
  - Mapas del departamento indicando despachos y empleados que pertenecen a estos despachos.

- Coordinación con herramientas de la universidad:
  - Universitas XXI: Para la información de publicaciones y proyectos existentes.
  - DSPACE: Para la información de publicaciones.

  Si estas herramientas pueden importar/ exportar información en formato, por ejemplo XML, sería relativamente sencillo modificar la aplicación para que importe/ exporte sus datos en el formato elegido. En la versión implementada ya se acepta importación de datos de publicaciones en formato BibteXML.

9. Referencias.

1.- [Wikipedia] **Fundación Wikimedia, Inc**

http://www.wikipedia.org

2.- [Java]: **Java.**
Java Network Technology. 2003.

http://java.sun.com

3.- [J2EE]: **Java 2 Enterprise Edition.**
Java 2 Enterprise Edition Network Technology. 2003.

http://java.sun.com/javaee/

4. - [Ruby on Rails]: **Ruby on Rails Framework.**
Ruby on Rails Community.

http://www.rubyonrails.org/

5. - [JBoss]: **JBoss.**
JBoss Community.

http://www.jboss.org

6. - [BEA Weblogic]: **BEA Weblogic Server 10.**

http://www.bea.com

7. - [WEBrick Server]: **WEBrick an HTTP server toolkit**.

http://www.**webrick**.org

8. - [Mongrel Server]: **Mongrel Trac**.

http://**mongrel**.rubyforge.org

9. - [LightTDP]: **Lighttpd fly light**.

http:// www.**lighttpd**.net

10. - [Struts]: **Struts Framework.**
Apache Struts. 2007.

http://www.jboss.org

11. - [Framework]: **Framework Comparision**.

http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

12. - [Spring]: **Spring Application Framework**.

http://www.**spring**framework.org

13. - [SQL Server]: **Microsoft SQL Server 2008**.

http://www.microsoft.com/spain/sql/default.mspx

14. - [MySQL]: **MySQL**. Sun Microsystems

http://www.**mysql**.com

15. - [PostgreSQL]: **PostgreSQL**.

www.**postgresql**.org

16. - [JRuby]: **JRuby.**

http://jruby.codehaus.org/

17.- **Página Web del Departamento de telemática.**

http://www.it.uc3m.es/

18.- **Foro de discursión para el modelo UML del proyecto.**

http://foros.it.uc3m.es/viewforum.php?f=20

19. - [Bibtext]: **BibTeXML standard.**

http://www.authopilot.com/xml/home.htm/

http://www.authopilot.com/xml/home.htm/

20. - [UML]: **Unified Modeleing Language.**

http://www.uml.org/

21. - [UML]: **Database Modeling in UML.**

http://www.sparxsystems.com.au/resources/uml_datamodel.html/

22. - [UML]: **A UML Profile for Data Modeling.**

http://www.agiledata.org/essays/umlDataModelingProfile.html/

23. - [UML]: **UML 2 Use CasesDiagrams.**

http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

http://www.pst.informatik.uni-muenchen.de/projekte/uwe/

24. - [UML]: **A UML Profile for Data Modeling.**

http://www.agiledata.org/essays/umlDataModelingProfile.html/

25. - [UML]: **Use Case templates. Alistair Cockburn.**

http://www.sparxsystems.com.au/resources/tutorial/use_case_model.html

26. - [UML]: **Use Case templates. Scenario Plus.**

http://www.scenarioplus.org.uk/download_uc_templates.html

27. - [UML]: **Use Case templates. IMS.**

http://www.imsproject.org/es/esv1p0/imses_usecasev1p0.html

28.- [Aptana]: **IDE Aptana Studio.**

http://www.aptana.com/studio

29. - [ruby/Ldap]: **RUBY/LDAP.**

http://ruby-ldap.sourceforge.net/

30. - [Ruby-J2EE]: **Comparativa de Zope, J2EE, Django and Ruby On Rails.**

http://oodt.jpl.nasa.gov/better-web-app.mov

32. - [Joomla!]: **Joomla.**

http://www.agiledata.org/essays/umlDataModelingProfile.html/

33. - [Drupal]: **Drupal.**

http://en.wikipedia.org/wiki/Drupal

34. - [CMS]: **Content Management Systems.**

http://en.wikipedia.org/wiki/List_of_content_management_systems

10. Anéxos

A continuación se mostrará una serie de documentos que ayudarán a la comprensión de este proyecto y su lugar dentro de un conjunto de aplicaciones desarrolladas para crear un framework que permite el control total sobre el contenido de la Web del departamento de telemática.

Esta sección consta de los siguientes puntos:

- Anéxo A
  - Modelo de Datos
    - Modelo Conceptual (Diagrama de Clases Principal).
    - Modelo conceptual (Diagrama de Clases singulares).
    - OCL (Restrincciones aceptadas y desechadas).
    - Modelo Físico
    - Modelo de Realización A
    - Modelo de Realización B
  - Script de Base deDatos
- Anéxo B
  - Casos de Uso Extendidos: Templates
- Anéxo C
  - Montando Ruby on Rails en Debian

10.1 Anéxo A.

Este apéndice muestra tanto el modelo de datos como el script de generación de Base de Datos creado por **D. Carlos Holgado Molinillo** que fue evaluado por los directores del proyecto de desarrollo de este sistema de Web Departamental.

Ambos documentos fueron íntegramente desarrollados por **D. Carlos Holgado Molinillo** y se muestran como apéndices de la memoria de **D. Adolfo Miguel Catalán García-Manso** para posibilitar la comprensión del diseño global del sistema y las tecnologías empleadas en él.

10.1.1 Modelo de Datos.

Un **modelo de datos** es un lenguaje orientado a describir una b.b.d.d. Típicamente un Modelo de Datos permite describir:

- Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

Otro enfoque es pensar que un **modelo de datos** permite describir los elementos que intervienen en una realidad problema dado y la forma en que se relacionan esos elementos entre sí.

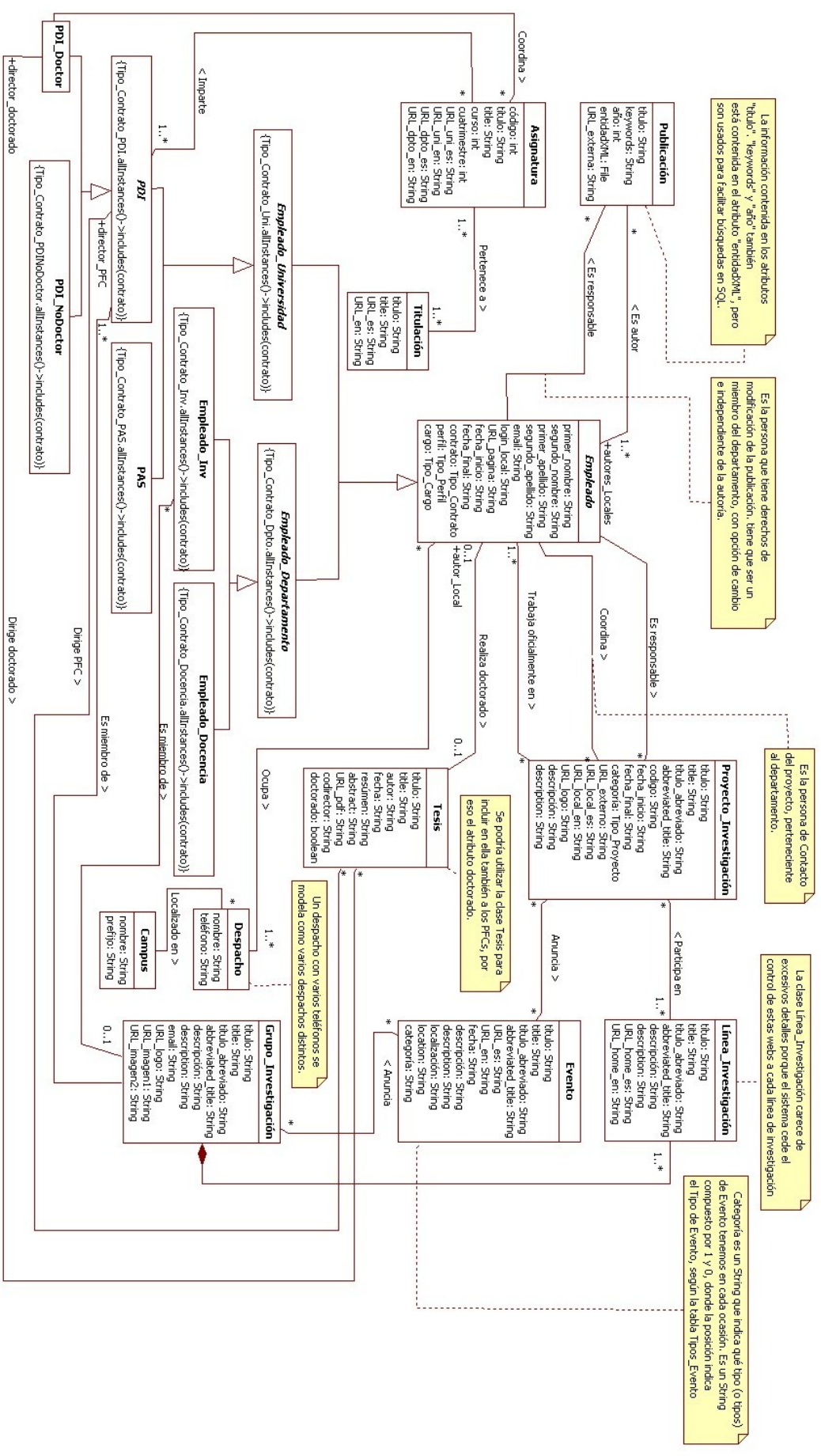
No hay que perder de vista que una b.b.d.d. siempre está orientada a resolver un problema determinado, por lo que los dos enfoques propuestos son necesarios en cualquier desarrollo de software.

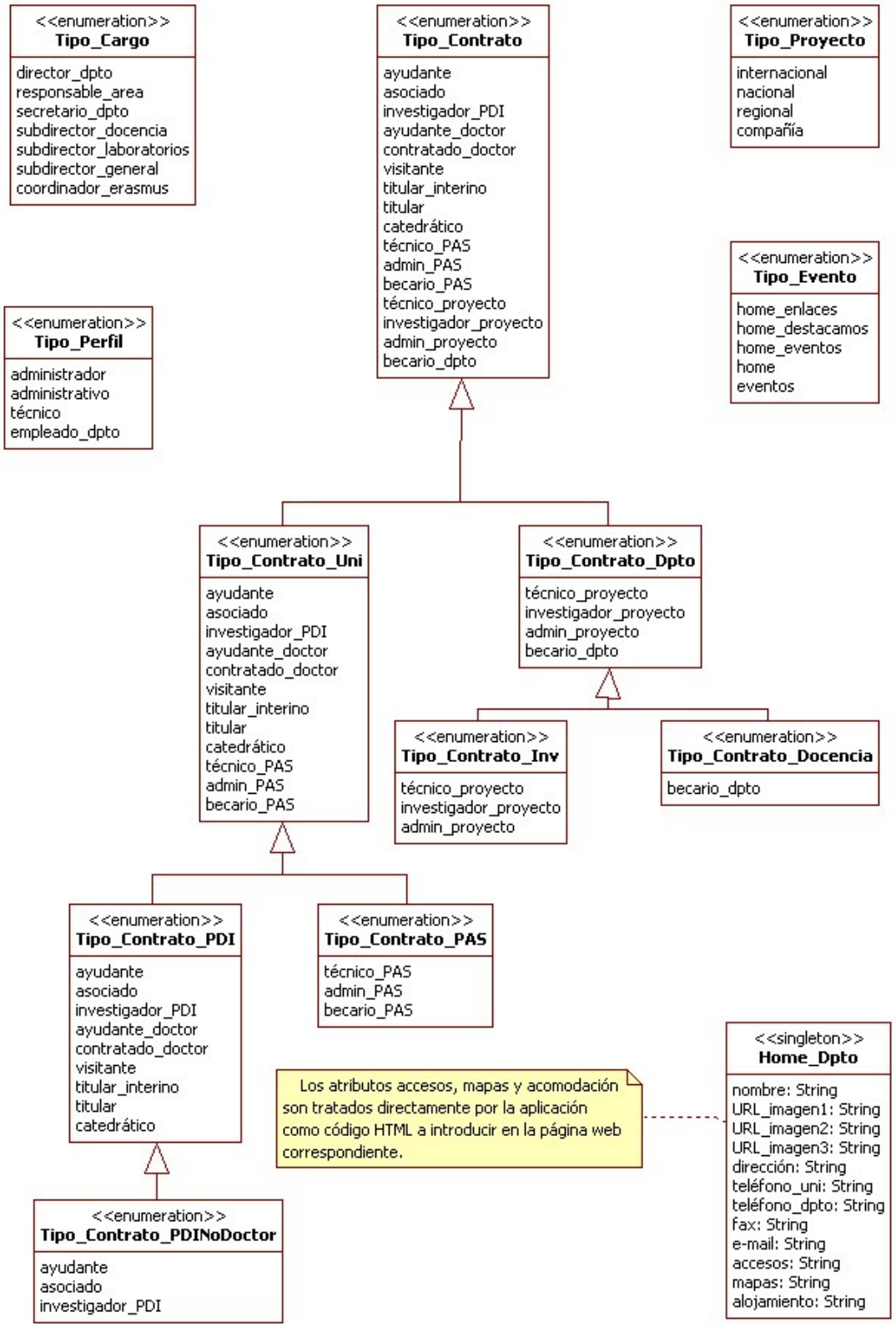
10.1.1.1 Modelo Conceptual.

Durante esta actividad, se construye un esquema conceptual que representa objetos, sus relaciones y colaboraciones que existen en el dominio designado. En aplicaciones de hypermedia convencionales, es decir, aquellos en los que los componentes de la hypermedia no serán modificados durante su ejecución, se podría usar un modelo semántico estructural [Banerjee87], sin embargo, cuando la base de información puede cambiar dinámicamente o si se piensa realizar cómputos complejos o consultas en los objetos o el esquema, se necesita una abundante conducta del modelo orientado a objetos.

El esquema conceptual es construido en las clases, relaciones y sub-sistemas. Las clases son descritas como de costumbre en el modelo orientado a objetos, sin embargo, pueden multi-digitar atributos representando perspectivas diferentes de la misma entidad del mundo. Se usa una notación que es similar a UML [OMG 00], la Clase y Tarjetas de las relaciones, similar a las tarjetas de CRC [Wirfs-Brock 90] son usadas como una ayuda de la documentación, ayudando remontar decisiones de diseño enviados y al revés.

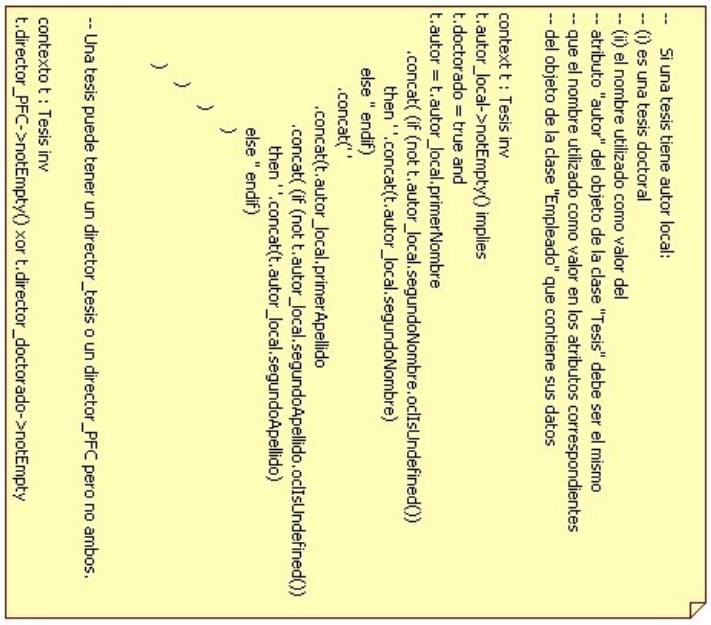
Las metodologías orientadas a objetos y bibliografía en el área. Sin embargo hay algunas decisiones modeladas que aparecen en cualquier proceso en el que puede impactar en la estructura navegacional de aplicaciones de la hypermedia. En este papel, se enfoca en las decisiones de diseño que afectan tal estructura.

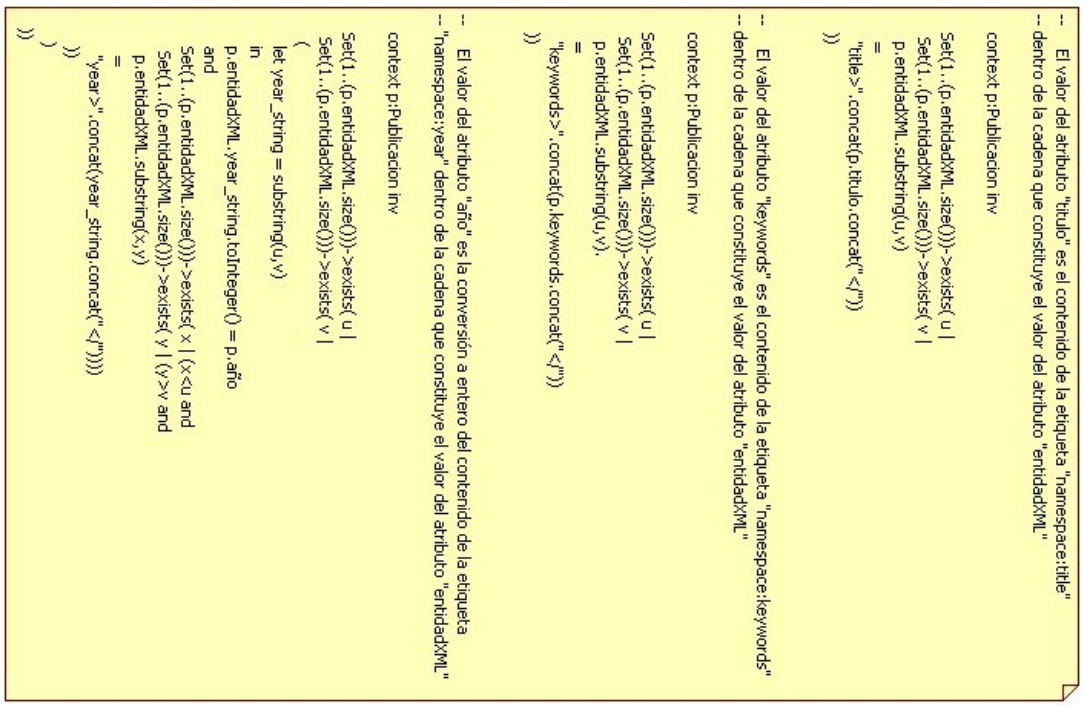

La información contenida en los atributos "título", "keywords" y "año" también está contenida en el atributo "entidadXML", pero son usados para facilitar búsquedas en SQL.
Es la persona que tiene derechos de modificación de la publicación, tiene que ser un miembro del departamento, con opción de cambio e independiente de la autoría.
Es la persona de Contacto del proyecto, perteneciente al departamento.
La clase Línea_Investigación carece de excesivos detalles porque el sistema cede el control de estas webs a cada línea de investigación
Categoría es un String que indica qué tipo (o tipos) de Evento tenemos en cada ocasión. Es un String compuesto por 1 y 0, donde la posición indica el Tipo de Evento, según la tabla Tipos_Evento
Se podría utilizar la clase Tesis para incluir en ella también a los PFCs, por eso el atributo doctorado.
Un despacho con varios teléfonos se modela como varios despachos distintos.
Publicación
título: String
keywords: String
año: int
entidadXML: File
URL_externa: String
Asignatura
código: int
título: String
title: String
curso: int
cuatrimestre: int
URL_uni_es: String
URL_uni_en: String
URL_dpto_es: String
URL_dpto_en: String
Titulación
título: String
title: String
URL_es: String
URL_en: String
Empleado
primer_nombre: String
segundo_nombre: String
primer_apellido: String
segundo_apellido: String
email: String
login_local: String
URL_pagina: String
fecha_inicio: String
fecha_final: String
contrato: Tipo_Contrato
perfil: Tipo_Perfil
cargo: Tipo_Cargo
Proyecto_Investigación
título: String
title: String
título_abreviado: String
abbreviated_title: String
código: String
fecha_inicio: String
fecha_final: String
categoría: Tipo_Proyecto
URL_externo: String
URL_local_es: String
URL_local_en: String
URL_logo: String
descripción: String
description: String
Línea_Investigación
título: String
title: String
título_abreviado: String
abbreviated_title: String
descripción: String
description: String
URL_home_es: String
URL_home_en: String
Evento
título: String
title: String
título_abreviado: String
abbreviated_title: String
URL_es: String
URL_en: String
fecha: String
descripción: String
description: String
localización: String
location: String
categoría: String
Tesis
título: String
title: String
autor: String
fecha: String
resúmen: String
abstract: String
URL_pdf: String
codirector: String
doctorado: boolean
Despacho
nombre: String
teléfono: String
Campus
nombre: String
prefijo: String
Grupo_Investigación
título: String
title: String
título_abreviado: String
abbreviated_title: String
descripción: String
description: String
email: String
URL_logo: String
URL_imagen1: String
URL_imagen2: String
Empleado_Universidad
{Tipo_Contrato_Uni.allInstances()->includes(contrato)}
Empleado_Departamento
{Tipo_Contrato_Dpto.allInstances()->includes(contrato)}
Empleado_Inv
{Tipo_Contrato_Inv.allInstances()->includes(contrato)}
Empleado_Docencia
{Tipo_Contrato_Docencia.allInstances()->includes(contrato)}
PAS
{Tipo_Contrato_PAS.allInstances()->includes(contrato)}
PDI
{Tipo_Contrato_PDI.allInstances()->includes(contrato)}
PDI_Doctor
PDI_NoDoctor
{Tipo_Contrato_PDINoDoctor.allInstances()->includes(contrato)}
Coordina >
< Es responsable
< Es autor
Pertenece a >
+autores_Locales
Es responsable >
Coordina >
Trabaja oficialmente en >
Realiza doctorado >
0..1
+autor_Local
< Participa en
Anuncia >
< Anuncia
Ocupa >
Localizado en >
< Imparte
+director_PFC
Dirige PFC >
Es miembro de >
Dirige doctorado >
+director_doctorado


<<enumeration>>
Tipo_Cargo
director_dpto
responsable_area
secretario_dpto
subdirector_docencia
subdirector_laboratorios
subdirector_general
coordinador_erasmus
<<enumeration>>
Tipo_Contrato
ayudante
asociado
investigador_PDI
ayudante_doctor
contratado_doctor
visitante
titular_interino
titular
catedrático
técnico_PAS
admin_PAS
becario_PAS
técnico_proyecto
investigador_proyecto
admin_proyecto
becario_dpto
<<enumeration>>
Tipo_Proyecto
internacional
nacional
regional
compañía
<<enumeration>>
Tipo_Evento
home_enlaces
home_destacamos
home_eventos
home
eventos
<<enumeration>>
Tipo_Perfil
administrador
administrativo
técnico
empleado_dpto
<<enumeration>>
Tipo_Contrato_Uni
ayudante
asociado
investigador_PDI
ayudante_doctor
contratado_doctor
visitante
titular_interino
titular
catedrático
técnico_PAS
admin_PAS
becario_PAS
<<enumeration>>
Tipo_Contrato_Dpto
técnico_proyecto
investigador_proyecto
admin_proyecto
becario_dpto
<<enumeration>>
Tipo_Contrato_Inv
técnico_proyecto
investigador_proyecto
admin_proyecto
<<enumeration>>
Tipo_Contrato_Docencia
becario_dpto
<<enumeration>>
Tipo_Contrato_PDI
ayudante
asociado
investigador_PDI
ayudante_doctor
contratado_doctor
visitante
titular_interino
titular
catedrático
<<enumeration>>
Tipo_Contrato_PAS
técnico_PAS
admin_PAS
becario_PAS
Los atributos accesos, mapas y acomodación son tratados directamente por la aplicación como código HTML a introducir en la página web correspondiente.
<<singleton>>
Home_Dpto
nombre: String
URL_imagen1: String
URL_imagen2: String
URL_imagen3: String
dirección: String
teléfono_uni: String
teléfono_dpto: String
fax: String
e-mail: String
accesos: String
mapas: String
alojamiento: String
<<enumeration>>
Tipo_Contrato_PDINoDoctor
ayudante
asociado
investigador_PDI

10.1.1.2 OCL.

OCL2.0 (Object Constraint Language 2.0) fue adoptado en octubre de 2003 por el grupo OMG como parte de UML 2.0. OCL es un lenguaje para la descripción formal de expresiones en los modelos UML. Sus expresiones pueden representar *invariantes*, *precondiciones*, *postcondiciones*, *inicializaciones*, *guardias*, *reglas de derivación*, así como *consultas* a objetos para determinar sus condiciones de estado. Se trata de un lenguaje sin efectos de borde, de manera que la verificación de una condición, que se presupone una operación instantánea, nunca altera los objetos del modelo. Su papel principal es el de completar los diferentes artefactos de la notación UML con requerimientos formalmente expresados.





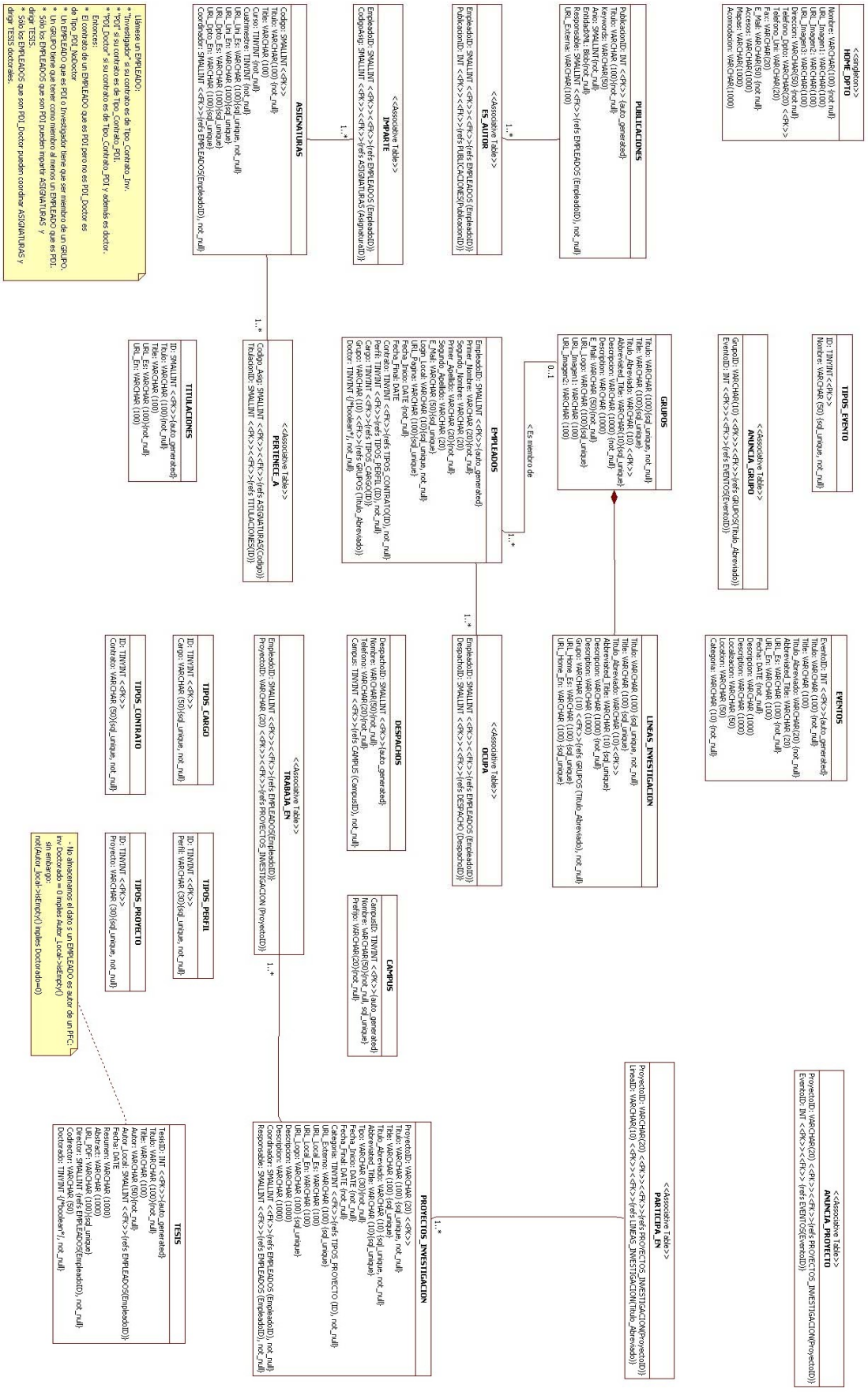
```
--    Si una publicación tiene autores locales, el nombre utilizado en el valor
-- del atributo "entidadXML" del objeto de la clase "Publicación"
-- debe ser el mismo que el nombre utilizado como valor en los atributos
-- correspondientes del objeto de la clase "Empleado" que
-- contiene sus datos, para cada uno de los autores locales


context p : Publicación inv

p.autores_locales->(forall x |
   let full_name = x.primerNombre
         .concat( (if (not x.segundoNombre.oclIsUndefined())
               then ' '.concat(x.segundoNombre)
            else '' endif)
               .concat(' '
                  .concat(x.primerApellido
                     .concat( (if (not x.segundoApellido.oclIsUndefined())
                           then ' '.concat(x.segundoApellido)
                        else '' endif)
                     )
                  )
               )
         )
   in
   ( Set(1..(p.entidadXML.size()))->exists (y |
      Set(1..(p.entidadXML.size()))->exists (z |
         p.entidadXML.substring(y,z) = full_name ) ) )
```

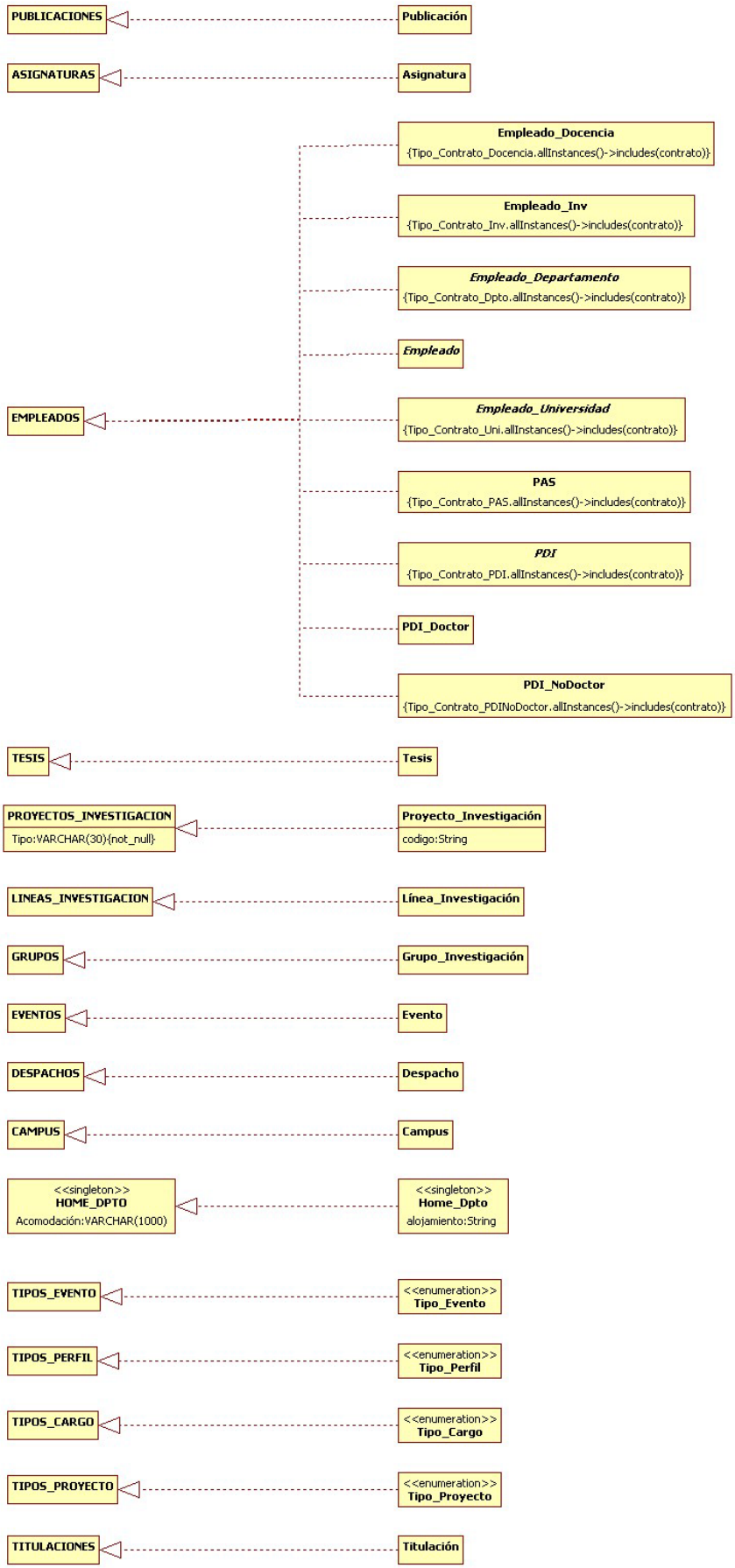
10.1.1.3 Modelo Físico.

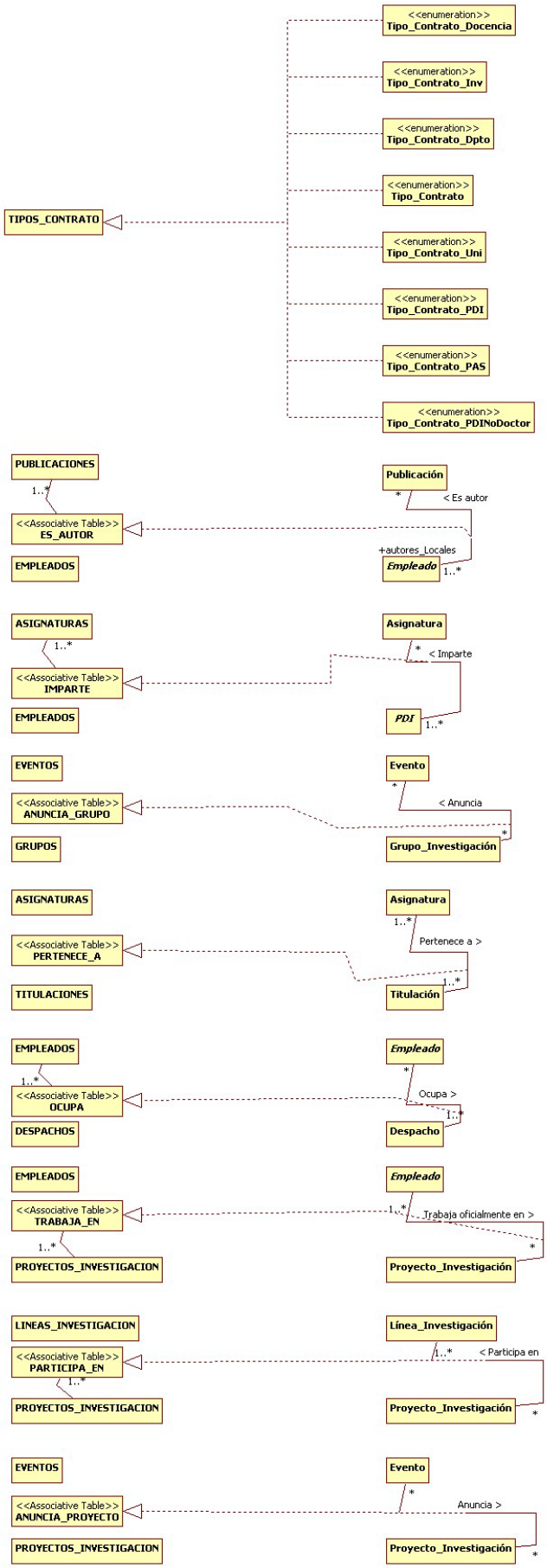
El modelo de datos UML físico permite comprender la implementación del modelo conceptual en la base de datos relacional. Es necesario comprender que el modelo físico no sustituye al modelo conceptual en nuestro diseño, y que incluso dispone de menos información y restricciones que el modelo conceptual, a pesar de no ser así normalmente.


<<singleton>>
HOME_DPTO
Nombre: VARCHAR(100) {not null}
URL_Imagen1: VARCHAR(100)
URL_Imagen2: VARCHAR(100)
URL_Imagen3: VARCHAR(100)
Dirección: VARCHAR(100) {not null}
Teléfono_Dpto: VARCHAR(20) <<PK>>
Teléfono_Uni: VARCHAR(20)
Fax: VARCHAR(20)
E_Mail: VARCHAR(50) {not null}
Acceso: VARCHAR(1000)
Mapas: VARCHAR(1000)
Acomodacion: VARCHAR(1000)
PUBLICACIONES
PublicacionID: INT <<PK>> {auto_generated}
Título: VARCHAR (100){not_null}
Año: SMALLINT(4)
Anexo: VARCHAR(50)
Enlace2URL: BIG{not_null}
Responsable: SMALLINT <<FK>>{refs EMPLEADOS (EmpleadoID), not_null}
URL_Externa: VARCHAR(100)
<<Associative Table>>
ES_AUTOR
EmpleadoID: SMALLINT <<PK>><<FK>>{refs EMPLEADOS (EmpleadoID)}
PublicacionID: INT <<PK>><<FK>>{refs PUBLICACIONES(PublicacionID)}
1..*
1..*
ASIGNATURAS
Codigo: SMALLINT <<PK>>
Titulo: VARCHAR(100) {not_null}
Título: VARCHAR (100)
Curso: TINYINT {not_null}
Cuatrimestre: TINYINT {not_null}
URL_Uni_Es: VARCHAR (100){sql_unique, not_null}
URL_Uni_En: VARCHAR (100){sql_unique}
URL_Dpto_Es: VARCHAR (100){sql_unique}
URL_Dpto_En: VARCHAR (100){sql_unique}
Coordinador: SMALLINT <<FK>>{refs EMPLEADOS(EmpleadoID), not_null}
<<Associative Table>>
IMPARTE
EmpleadoID: SMALLINT <<PK>><<FK>>{refs EMPLEADOS (EmpleadoID)}
CodigoAsig: SMALLINT <<PK>><<FK>>{refs ASIGNATURAS (AsignaturaID)}
1..*
GRUPOS
Título: VARCHAR (100){sql_unique, not_null}
Título: VARCHAR (100) {sql_unique}
Título_Abreviado: VARCHAR (10) <<PK>>
Abbreviated_Title: VARCHAR(10){sql_unique}
Descripcion: VARCHAR (1000) {not_null}
Description: VARCHAR (1000)
E_Mail: VARCHAR (50){not_null}
Logo: VARCHAR (100){sql_unique}
URL_Imagen1: VARCHAR (100)
URL_Imagen2: VARCHAR (100)
0..1
<<Es miembro de
1..*
<<Associative Table>>
ANUNCIA_GRUPO
GrupoID: VARCHAR(10) <<PK>><<FK>>{refs GRUPOS(Título_Abreviado)}
EventoID: INT <<PK>><<FK>>{refs EVENTOS(EventoID)}
TIPOS_EVENTO
ID: TINYINT <<PK>>
Nombre: VARCHAR (50) {sql_unique, not_null}
EVENTOS
EventoID: INT <<PK>>{auto_generated}
Título: VARCHAR (100) {not_null}
Título_Abreviado: VARCHAR(20) {not_null}
Abbreviated_Title: VARCHAR (20)
URL_Es: VARCHAR (100) {not_null}
URL_En: VARCHAR (100)
Descripcion: VARCHAR (1000)
Fecha: DATE {not_null}
Description: VARCHAR (1000)
Localizacion: VARCHAR (50)
Categorias: VARCHAR (10) {not_null}
EMPLEADOS
EmpleadoID: SMALLINT <<PK>>{auto_generated}
Primer_Nombre: VARCHAR (20){not_null}
Segundo_Nombre: VARCHAR (20)
Primer_Apellido: VARCHAR (20){not_null}
Segundo_Apellido: VARCHAR (20)
E_Mail: VARCHAR (50){sql_unique}
Login_Local: VARCHAR (10){sql_unique, not_null}
URL_Pagina: VARCHAR (100){sql_unique}
Fecha_Inicio: DATE {not_null}
Fecha_Fin: DATE
Contrato: TINYINT <<FK>>{refs TIPOS_CONTRATO(ID), not_null}
Perfil: TINYINT <<FK>>{refs TIPOS_PERFIL (ID), not_null}
Cargo: TINYINT <<FK>>{refs TIPOS_CARGO(ID)}
Grupo: VARCHAR (10) <<FK>>{refs GRUPOS (Título_Abreviado)}
Doctor: TINYINT {"Booleano"}, not_null}
<<Associative Table>>
OCUPA
EmpleadoID: SMALLINT <<PK>><<FK>>{refs EMPLEADOS (EmpleadoID)}
DespachoID: SMALLINT <<PK>><<FK>>{refs DESPACHO (DespachoID)}
LINEAS_INVESTIGACION
Título: VARCHAR (100) {sql_unique, not_null}
Título: VARCHAR (100) {sql_unique}
Título_Abreviado: VARCHAR (10) <<PK>>
Abbreviated_Title: VARCHAR (10) {sql_unique}
Descripcion: VARCHAR (1000) {not_null}
Description: VARCHAR (1000)
Grupo: VARCHAR (10) <<FK>>{refs GRUPOS (Título_Abreviado), not_null}
URL_Home_Es: VARCHAR (100) {sql_unique}
URL_Home_En: VARCHAR (100) {sql_unique}
<<Associative Table>>
ANUNCIA_PROYECTO
ProyectoID: VARCHAR(20) <<PK>><<FK>>{refs PROYECTOS_INVESTIGACION(ProyectoID)}
EventoID: INT <<PK>><<FK>>{refs EVENTOS(EventoID)}
<<Associative Table>>
PARTICIPA_EN
ProyectoID: VARCHAR(20) <<PK>><<FK>>{refs PROYECTOS_INVESTIGACION(ProyectoID)}
LineaID: VARCHAR(10) <<PK>><<FK>>{refs LINEAS_INVESTIGACION(Título_Abreviado)}
1..*
TITULACIONES
ID: SMALLINT <<PK>>{auto_generated}
Título: VARCHAR (100){not_null}
URL_Es: VARCHAR (100){not_null}
URL_En: VARCHAR (100)
<<Associative Table>>
PERTENECE_A
Codigo_Asig: SMALLINT <<PK>><<FK>>{refs ASIGNATURAS(Codigo)}
TitulacionID: SMALLINT <<PK>><<FK>>{refs TITULACIONES(ID)}
DESPACHOS
DespachoID: SMALLINT <<PK>>{auto_generated}
Número: VARCHAR(50){not_null}
Teléfono: VARCHAR(20){not_null}
Campus: TINYINT <<FK>>{refs CAMPUS (CampusID)}, not_null}
CAMPUS
CampusID: TINYINT <<PK>>{auto_generated}
Nombre: VARCHAR(50){not_null, sql_unique}
Prefijo: VARCHAR(20){not_null}
TIPOS_CARGO
ID: TINYINT <<PK>>
Cargo: VARCHAR (50){sql_unique, not_null}
TIPOS_PERFIL
ID: TINYINT <<PK>>
Perfil: VARCHAR (30){sql_unique, not_null}
TIPOS_CONTRATO
ID: TINYINT <<PK>>
Contrato: VARCHAR (50){sql_unique, not_null}
TIPOS_PROYECTO
ID: TINYINT <<PK>>
Proyecto: VARCHAR (30){sql_unique, not_null}
<<Associative Table>>
TRABAJA_EN
EmpleadoID: SMALLINT <<PK>><<FK>>{refs EMPLEADOS(EmpleadoID)}
ProyectoID: VARCHAR (20) <<PK>><<FK>>{refs PROYECTOS_INVESTIGACION (ProyectoID)}
1..*
PROYECTOS_INVESTIGACION
ProyectoID: VARCHAR (20) <<PK>>
Título: VARCHAR (100) {sql_unique, not_null}
Title: VARCHAR (100) {sql_unique}
Título_Abreviado: VARCHAR (10){sql_unique, not_null}
Abbreviated_Title: VARCHAR (10){sql_unique}
Categoria: TINYINT <<FK>>{refs TIPOS_PROYECTO (ID), not_null}
URL_Externo: VARCHAR (100) {sql_unique}
URL_Local_Es: VARCHAR (100)
URL_Local_En: VARCHAR (100)
URL_Logo: VARCHAR (100) {sql_unique}
Descripcion: VARCHAR (1000)
Description: VARCHAR (1000)
Fecha_Inicio: DATE {not_null}
Fecha_Final: DATE {not_null}
Categoria: TINYINT <<FK>>{refs TIPOS_PROYECTO (ID), not_null}
URL_Externo: VARCHAR (100) {sql_unique}
URL_Local_Es: VARCHAR (100)
URL_Local_En: VARCHAR (100)
URL_Logo: VARCHAR (100) {sql_unique}
Descripcion: VARCHAR (1000)
Coordinador: SMALLINT <<FK>>{refs EMPLEADOS (EmpleadoID), not_null}
Responsable: SMALLINT <<FK>>{refs EMPLEADOS (EmpleadoID), not_null}
TESIS
TesisID: INT <<PK>>{auto_generated}
Título: VARCHAR (100){not_null}
Title: VARCHAR (100)
Autor: VARCHAR (50){not_null}
Autor_Local: SMALLINT <<FK>>{refs EMPLEADOS(EmpleadoID)}
Fecha: DATE
Resumen: VARCHAR (1000)
Abstract: VARCHAR (1000)
URL_PDF: VARCHAR (100){sql_unique}
Director: SMALLINT {refs EMPLEADOS(EmpleadoID), not_null}
Codirector: VARCHAR (50)
Doctorado: TINYINT {"Booleano"}, not_null}
Llámese un EMPLEADO:
* "Investigador" si su contrato es de Tipo_Contrato_Inv.
* "PDI" si su contrato es de Tipo_Contrato_PDI.
* "PDI_Doctor" si su contrato es de Tipo_Contrato_PDI y además es doctor.
Entonces:
* El contrato de un EMPLEADO que es PDI pero no es PDI_Doctor es de Tipo_PDI_NDoctor
* Un EMPLEADO que es PDI o Investigador tiene que ser miembro de un GRUPO.
* Un GRUPO tiene que tener como miembro al menos un EMPLEADO que es PDI.
* Sólo los EMPLEADOS que son PDI_Doctor pueden impartir ASIGNATURAS y dirigir TESIS
* Sólo los EMPLEADOS que son PDI_Doctor pueden coordinar ASIGNATURAS y dirigir TESIS doctorales.
- No almacenamos el dato s un EMPLEADO es autor de un PFC:
inv: Doctorado = 0 implies Autor_Local->isEmpty()
sin embargo:
not(Autor_local->isEmpty()) implies Doctorado=0)

10.1.1.4 Modelo de Realización.

Una vez encontradas las entidades en el modelo conceptual, se pasará a traducir esta información a un modelo relacionado con la representación en b.b.d.d. Esta técnica de programación es conocida como Object-relational mapping (ORM), y los distintos problemas que pueden surgir son conocidos con el término general Object-relational impedance mismatch.


PUBLICACIONES
Publicación
ASIGNATURAS
Asignatura
Empleado_Docencia
{Tipo_Contrato_Docencia.allInstances()->includes(contrato)}
Empleado_Inv
{Tipo_Contrato_Inv.allInstances()->includes(contrato)}
Empleado_Departamento
{Tipo_Contrato_Dpto.allInstances()->includes(contrato)}
Empleado
EMPLEADOS
Empleado_Universidad
{Tipo_Contrato_Uni.allInstances()->includes(contrato)}
PAS
{Tipo_Contrato_PAS.allInstances()->includes(contrato)}
PDI
{Tipo_Contrato_PDI.allInstances()->includes(contrato)}
PDI_Doctor
PDI_NoDoctor
{Tipo_Contrato_PDINoDoctor.allInstances()->includes(contrato)}
TESIS
Tesis
PROYECTOS_INVESTIGACION
Tipo:VARCHAR(30){not_null}
Proyecto_Investigación
codigo:String
LINEAS_INVESTIGACION
Línea_Investigación
GRUPOS
Grupo_Investigación
EVENTOS
Evento
DESPACHOS
Despacho
CAMPUS
Campus
<<singleton>>
HOME_DPTO
Acomodación:VARCHAR(1000)
<<singleton>>
Home_Dpto
alojamiento:String
TIPOS_EVENTO
<<enumeration>>
Tipo_Evento
TIPOS_PERFIL
<<enumeration>>
Tipo_Perfil
TIPOS_CARGO
<<enumeration>>
Tipo_Cargo
TIPOS_PROYECTO
<<enumeration>>
Tipo_Proyecto
TITULACIONES
Titulación


<<enumeration>>
Tipo_Contrato_Docencia
<<enumeration>>
Tipo_Contrato_Inv
<<enumeration>>
Tipo_Contrato_Dpto
<<enumeration>>
Tipo_Contrato
TIPOS_CONTRATO
<<enumeration>>
Tipo_Contrato_Uni
<<enumeration>>
Tipo_Contrato_PDI
<<enumeration>>
Tipo_Contrato_PAS
<<enumeration>>
Tipo_Contrato_PDINoDoctor
PUBLICACIONES
1..*
<<Associative Table>>
ES_AUTOR
EMPLEADOS
Publicación
*
< Es autor
+autores_Locales
Empleado
1..*
ASIGNATURAS
1..*
<<Associative Table>>
IMPARTE
EMPLEADOS
Asignatura
*
< Imparte
PDI
1..*
EVENTOS
<<Associative Table>>
ANUNCIA_GRUPO
GRUPOS
Evento
*
< Anuncia
*
Grupo_Investigación
ASIGNATURAS
1..*
<<Associative Table>>
PERTENECE_A
TITULACIONES
Asignatura
1..*
Pertenece a >
1..*
Titulación
EMPLEADOS
1..*
<<Associative Table>>
OCUPA
DESPACHOS
Empleado
*
Ocupa >
1..*
Despacho
EMPLEADOS
<<Associative Table>>
TRABAJA_EN
1..*
PROYECTOS_INVESTIGACION
Empleado
1..*
Trabaja oficialmente en >
*
Proyecto_Investigación
LINEAS_INVESTIGACION
<<Associative Table>>
PARTICIPA_EN
1..*
PROYECTOS_INVESTIGACION
Línea_Investigación
1..*
< Participa en
Proyecto_Investigación
*
EVENTOS
<<Associative Table>>
ANUNCIA_PROYECTO
PROYECTOS_INVESTIGACION
Evento
*
Anuncia >
Proyecto_Investigación
*

10.1.2 Script de Base de Datos.

```
□□ MySQL Admini s t rator dump 1 . 4
□□
□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□ Se rve r v e r s i on 4.1.22□community□nt
/_!40101 SET @OLD CHARACTER SET CLIENT=@@CHARACTER SET CLIENT _/;
/_!40101 SET @OLD CHARACTER SET RESULTS=@@CHARACTER SET RESULTS _/;
/_!40101 SET @OLD COLLATION CONNECTION=@@COLLATION CONNECTION _/;
/_!40101 SET NAMES ut f 8 _/;
/_!40014 SET @OLD UNIQUE CHECKS=@@UNIQUE CHECKS, UNIQUE CHECKS=0 _/;
/_!40014 SET @OLD FOREIGN KEY CHECKS=@@FOREIGN KEY CHECKS, FOREIGN KEY CHECKS=0 _/;
/_!40101 SET @OLD SQL MODE=@@SQLMODE, SQL MODE='NO AUTO VALUE ON ZERO' _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` anuncia grupo `
□□
DROP TABLE IF EXISTS `ANUNCIA GRUPO` ;
CREATE TABLE `ANUNCIA GRUPO` (
`GrupoID ` varchar (10) NOT NULL d e f a u l t ' ' ,
`EventoID ` i n t (10) uns igned NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( `GrupoID ` , ` EventoID ` ) ,
KEY ` FK anuncia grupo 2 ` ( ` EventoID ` ) ,
CONSTRAINT ` FK anuncia grupo 1 ` FOREIGN KEY ( `GrupoID ` ) REFERENCES `GRUPOS` ( ` Ti tulo Abr eviado
` ) ON UPDATE CASCADE,
CONSTRAINT ` FK anuncia grupo 2 ` FOREIGN KEY ( ` EventoID ` ) REFERENCES `EVENTOS` ( ` EventoID ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( `Grupo ` ) REFER `webdpto/
GRUPOS` ( ` Ti tulo Abr eviado ` ) ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` anuncia grupo `
□□
/_!40000 ALTER TABLE `ANUNCIA GRUPO` DISABLE KEYS _/;
INSERT INTO `ANUNCIA GRUPO` VALUES ( 'GAST' , 8 ) ;
/_!40000 ALTER TABLE `ANUNCIA GRUPO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` anunc ia proye c to `
□□
DROP TABLE IF EXISTS `ANUNCIA PROYECTO` ;
CREATE TABLE `ANUNCIA PROYECTO` (
` ProyectoID ` varchar (20) NOT NULL d e f a u l t ' ' ,
`EventoID ` i n t (10) uns igned NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( ` ProyectoID ` , ` EventoID ` ) ,
KEY ` FK anuncia proyecto 2 ` ( ` EventoID ` ) ,
CONSTRAINT ` FK anuncia proyecto 1 ` FOREIGN KEY ( ` ProyectoID ` ) REFERENCES `
PROYECTOS INVESTIGACION` ( ` ProyectoID ` ) ON UPDATE CASCADE,
CONSTRAINT ` FK anuncia proyecto 2 ` FOREIGN KEY ( ` EventoID ` ) REFERENCES `EVENTOS` ( ` EventoID ` )
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` ProyectoID ` ) REFER `
webdpto/PROYECTOS INVESTIGACION' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` anunc ia proye c to `
□□
/_!40000 ALTER TABLE `ANUNCIA PROYECTO` DISABLE KEYS _/;
INSERT INTO `ANUNCIA PROYECTO` VALUES ( ' 3 4 2 3 4 2 3 ' , 8 ) ;
/_!40000 ALTER TABLE `ANUNCIA PROYECTO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` as igna tur as `
□□
DROP TABLE IF EXISTS `ASIGNATURAS` ;
CREATE TABLE `ASIGNATURAS` (
`Codigo ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
` Ti tulo ` varchar (50) NOT NULL d e f a u l t ' ' ,
` Ti t l e ` varchar (50) d e f a u l t NULL,
`Curso ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
` Cuat r imes t re ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
`URL Uni Es ` varchar (50) NOT NULL d e f a u l t ' ' ,
`URL Uni En ` varchar (50) d e f a u l t NULL,
`URL Dpto Es ` varchar (50) d e f a u l t NULL,
`URL Dpto En ` varchar (50) d e f a u l t NULL,
` Coordinador ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( ` Codigo ` ) ,
KEY ` FK as ignaturas 1 ` ( ` Coordinador ` ) ,
CONSTRAINT ` FK as ignaturas 1 ` FOREIGN KEY ( ` Coordinador ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID
` ) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` Coordinador ` ) REFER `
webdpto/ empleados ' ;
```

```
□□
□□ Dumping data f o r t abl e `webdpto ` . ` as ignaturas `
□□
/_!40000 ALTER TABLE `ASIGNATURAS` DISABLE KEYS _/;
INSERT INTO `ASIGNATURAS` VALUES (2334 , ' as igna2 ' , ' as igna2 ' , 2 , 2 , ' as igna2 ' , ' as igna2 ' , ' as igna2 ' , '
as igna2 ' , 8 ) ;
INSERT INTO `ASIGNATURAS` VALUES (10904 , ' Sof tware de comunicaciones ' , ' Communication Sof tware
' , 5 , 1 , ' ht tp : / /www3. uc3m. e s / r e ina /Fichas / f i c h a s 1 / 0 7 1 0 9 0 4 . ' , ' ht tp : / /www. i t . uc3m. e s / s p i c k i n /
doc enc ia / comsof t / ind ' , ' ht tp : / /www3. uc3m. e s / r e ina /Fichas / f i c h a s 2 / 0 7 1 0 9 0 4 . ' , ' ht tp : / /www. i t .
uc3m. e s / s p i c k i n / doc enc ia / comsof t / ind ' , 9 ) ;
INSERT INTO `ASIGNATURAS` VALUES (34233 , ' adsd ' , ' asdasd ' , 1 , 0 , ' asdas ' , ' asd ' , ' dasd ' , ' asdsa ' , 2 ) ;
/_!40000 ALTER TABLE `ASIGNATURAS` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` campus `
□□
DROP TABLE IF EXISTS `CAMPUS` ;
CREATE TABLE `CAMPUS` (
`CampusID` sma l l i n t ( 6 ) uns igned NOT NULL auto increment ,
`Nombre ` varchar (50) NOT NULL d e f a u l t '' ,
` Pr e f i j o ` varchar (20) NOT NULL d e f a u l t '' ,
PRIMARY KEY ( `CampusID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` campus `
□□
/_!40000 ALTER TABLE `CAMPUS` DISABLE KEYS _/;
INSERT INTO `CAMPUS` VALUES ( 2 , ' Legan_es ' , ' (+3 4 ) 91□733 ') ;
INSERT INTO `CAMPUS` VALUES ( 3 , ' Colmenarejo ' , ' (+3 4 ) 91□736 ') ;
INSERT INTO `CAMPUS` VALUES ( 4 , ' Getafe ' , ' (+3 4 ) 91□738 ') ;
/_!40000 ALTER TABLE `CAMPUS` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` despacho `
□□
DROP TABLE IF EXISTS `DESPACHO` ;
CREATE TABLE `DESPACHO` (
`DespachoID ` sma l l i n t ( 5 ) uns igned NOT NULL auto increment ,
`Nombre ` varchar (50) NOT NULL d e f a u l t '' ,
` Telefono ` varchar (20) NOT NULL d e f a u l t '' ,
`Campus ` sma l l i n t ( 6 ) uns igned NOT NULL d e f a u l t '0 ' ,
PRIMARY KEY ( ` DespachoID ` ) ,
KEY ` FK despacho 1 ` ( `Campus ` ) ,
CONSTRAINT ` FK despacho 1 ` FOREIGN KEY ( `Campus ` ) REFERENCES ` campus ` ( `CampusID ` ) ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 9216 kB; ( `Campus ` ) REFER `webdpto/
CAMPUS` ( ` CampusID ` ) ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` despacho `
□□
/_!40000 ALTER TABLE `DESPACHO` DISABLE KEYS _/;
INSERT INTO `DESPACHO` VALUES ( 1 , 'A. 1 . 1 . 1 ' , ' 3 3 4 5 6 7 ' , 3 ) ;
INSERT INTO `DESPACHO` VALUES ( 2 , 'A. 1 . 1 . 2 ' , ' 3 3 4 5 6 ' , 2 ) ;
INSERT INTO `DESPACHO` VALUES ( 3 , 'A. 1 . 1 . 4 ' , ' 3 4 6 5 4 ' , 4 ) ;
INSERT INTO `DESPACHO` VALUES ( 4 , 'B. 1 . 1 . 2 ' , ' 3 4 5 3 4 5 ' , 3 ) ;
/_!40000 ALTER TABLE `DESPACHO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` empleados `
□□
DROP TABLE IF EXISTS `EMPLEADOS` ;
CREATE TABLE `EMPLEADOS` (
`EmpleadoID ` sma l l i n t ( 5 ) uns igned NOT NULL auto increment ,
` Primer Nombre ` varchar (20) NOT NULL d e f a u l t '' ,
` Segundo Nombre ` varchar (20) d e f a u l t NULL,
` Pr ime r Ape l l ido ` varchar (20) NOT NULL d e f a u l t '' ,
` Segundo Apel l ido ` varchar (20) d e f a u l t NULL,
` E Mai l ` varchar (50) d e f a u l t NULL,
` Login Local ` varchar (10) NOT NULL d e f a u l t '' ,
`URL Pagina ` varchar (50) d e f a u l t NULL,
` Fe cha Ini c i o ` date NOT NULL d e f a u l t '0000□00□00 ' ,
` Fecha Final ` date d e f a u l t NULL,
` Contrato ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0 ' ,
` Pe r f i l ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0 ' ,
`Cargo ` t i n y i n t ( 3 ) uns igned d e f a u l t '0 ' ,
`Grupo ` varchar (10) d e f a u l t '' ,
`Doctor ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0 ' ,
PRIMARY KEY ( ` EmpleadoID ` ) ,
KEY ` FK empleados 1 ` ( ` Contrato ` ) ,
KEY ` FK empleados 2 ` ( ` Pe r f i l ` ) ,
KEY ` FK empleados 3 ` ( ` Cargo ` ) ,
KEY ` FK empleados 4 ` ( `Grupo ` ) ,
CONSTRAINT ` FK empleados 1 ` FOREIGN KEY ( ` Contrato ` ) REFERENCES `TIPOS CONTRATO` ( ` ID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK empleados 2 ` FOREIGN KEY ( ` Pe r f i l ` ) REFERENCES `TIPOS PERFIL ` ( ` ID ` ) ON UPDATE
CASCADE,
```

```
CONSTRAINT ` FK empleados 3 ` FOREIGN KEY ( ` Cargo ` ) REFERENCES `TIPOS CARGO` ( ` ID ` ) ON UPDATE
CASCADE,
CONSTRAINT ` FK empleados 4 ` FOREIGN KEY ( `Grupo ` ) REFERENCES `GRUPOS` ( ` Ti tulo Abr eviado ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; InnoDB f r e e : 3072 kB; ( `
Contrato ` ) REF' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` empleados `
□□
/_!40000 ALTER TABLE `EMPLEADOS` DISABLE KEYS _/;
INSERT INTO `EMPLEADOS` VALUES ( 1 , ' Juan ' , ' Diego ' , ' Perz ' , ' Garcia ' , ' diego@uc3m . es ' , ' diego ' , ' cvb
' , '2008□03□20 ' ,NULL, 2 , 1 , 2 , 'GAST' , 1 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 2 , ' Sara ' , ' ' , ' Gonzalez ' , ' Peron ' , ' sara@uc3m . es ' , ' sara ' , 'www
' , '2008□03□20 ' ,NULL, 5 , 2 , 3 , 'NETCOM' , 1 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 3 , 'Ana ' , ' ' , ' Garcia ' , ' Jul ian ' , ' ana@uc3m. es ' , ' ana ' , ' f f
' , '2008□03□20 ' ,NULL, 2 , 2 , 6 ,NULL, 0 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 4 , ' sant iago ' , ' Garcia ' , ' Garcia ' , ' Perez ' , ' santiago@uc3m . es ' , '
sant iago ' , ' f g f g ' , '2008□03□20 ' ,NULL, 5 , 3 , 6 , 'NETCOM' , 1 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 8 , ' Prof e sor ' , ' Prof e sor ' , ' Prof e sor ' , ' Prof e sor ' , ' profesor@uc3m .
es ' , ' pr o f e s o r ' , ' sdf ' , '2008□03□23 ' ,NULL, 5 , 2 ,NULL, 'NETCOM' , 0 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 9 , ' t e cni co ' , ' t e cni c o ' , ' t e cni c o ' , ' t e cni c o ' , ' tecnico@uc3m . es ' , '
t e cni c o ' , ' s d f s d f s ' , '2008□04□04 ' ,NULL, 5 , 2 , 1 , 'NETCOM' , 0 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 1 0 , ' admini s t r a t ivo ' , ' admini s t r a t ivo ' , ' admini s t r a t ivo ' , '
admini s t r a t ivo ' , ' administrativo@uc3m . es ' , ' admini s t ra ' , ' sdf sdf ' , '2008□04□04 ' ,NULL, 5 , 3 , 3 , '
NETCOM' , 0 ) ;
INSERT INTO `EMPLEADOS` VALUES ( 1 1 , ' admin ' , ' admin ' , ' admin ' , ' admin ' , ' admin@uc3m. es ' , ' admin ' , '
f d g d f g t g t j j ' , '2008□04□04 ' ,NULL, 5 , 1 , 2 , 'NETCOM' , 1 ) ;
/_!40000 ALTER TABLE `EMPLEADOS` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` e s aut o r `
□□
DROP TABLE IF EXISTS `ES AUTOR` ;
CREATE TABLE `ES AUTOR` (
`EmpleadoID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
` Publ icacionID ` i n t (10) uns igned NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( ` EmpleadoID ` , ` Publ icacionID ` ) ,
KEY ` FK es autor 2 ` ( ` Publ icacionID ` ) ,
CONSTRAINT ` FK es autor 1 ` FOREIGN KEY ( ` EmpleadoID ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK es autor 2 ` FOREIGN KEY ( ` Publ icacionID ` ) REFERENCES `PUBLICACIONES` ( `
Publ icacionID ` ) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` EmpleadoID ` ) REFER `
webdpto/ empleados ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` e s aut o r `
□□
/_!40000 ALTER TABLE `ES AUTOR` DISABLE KEYS _/;
INSERT INTO `ES AUTOR` VALUES ( 1 , 1 ) ;
INSERT INTO `ES AUTOR` VALUES ( 2 , 1 ) ;
INSERT INTO `ES AUTOR` VALUES ( 3 , 1 ) ;
INSERT INTO `ES AUTOR` VALUES ( 3 , 2 ) ;
INSERT INTO `ES AUTOR` VALUES ( 8 , 2 ) ;
/_!40000 ALTER TABLE `ES AUTOR` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` eventos `
□□
DROP TABLE IF EXISTS `EVENTOS` ;
CREATE TABLE `EVENTOS` (
`EventoID ` i n t (10) uns igned NOT NULL auto increment ,
` Ti tulo ` varchar (100) NOT NULL d e f a u l t ' ' ,
` Ti t l e ` varchar (100) d e f a u l t NULL,
` Ti tulo Abr eviado ` varchar (20) NOT NULL d e f a u l t ' ' ,
` Abbr eviat ed Ti t l e ` varchar (20) d e f a u l t NULL,
`URL Es ` varchar (50) NOT NULL d e f a u l t ' ' ,
`URL En ` varchar (50) d e f a u l t NULL,
` Fecha ` date NOT NULL d e f a u l t '0000□00□00 ' ,
` De s c r ipc ion ` varchar (1000) ,
` De s c r ipt ion ` varchar (1000) ,
` Lo c a l i z a c i on ` varchar (50) d e f a u l t NULL,
` Locat ion ` varchar (50) d e f a u l t NULL,
` Categor ia ` varchar (10) NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( ` EventoID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` Categor ia ` ) REFER `webdpto
/ t ipo s even ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` eventos `
□□
/_!40000 ALTER TABLE `EVENTOS` DISABLE KEYS _/;
INSERT INTO `EVENTOS` VALUES ( 8 , ' sap ' , ' sap ' , ' sap ' , ' sap ' , ' sap ' , ' sap ' , '2008□03□21 ' , ' sap ' , ' sap ' , '
sap ' ,NULL, ' 1 1 0 1 1 0 0 0 0 ' ) ;
/_!40000 ALTER TABLE `EVENTOS` ENABLE KEYS _/;
```

```
□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` grupos `
□□
DROP TABLE IF EXISTS `GRUPOS` ;
CREATE TABLE `GRUPOS` (
` Ti tulo ` varchar (50) NOT NULL d e f a u l t '' ,
` Ti t l e ` varchar (50) d e f a u l t NULL,
` Ti tulo Abr eviado ` varchar (10) NOT NULL d e f a u l t '' ,
` Abbr eviat ed Ti t l e ` varchar (10) d e f a u l t NULL,
` De s c r ipc ion ` varchar (1000) NOT NULL,
` De s c r ipt ion ` varchar (1000) ,
` E Mai l ` varchar (50) NOT NULL d e f a u l t '' ,
`URL Logo ` varchar (50) d e f a u l t NULL,
`URL Imagen1 ` varchar (50) d e f a u l t NULL,
`URL Imagen2 ` varchar (50) d e f a u l t NULL,
PRIMARY KEY ( ` Ti tulo Abr eviado ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` grupos `
□□
/_!40000 ALTER TABLE `GRUPOS` DISABLE KEYS _/;
INSERT INTO `GRUPOS` VALUES ( 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , '
GAST' ) ;
INSERT INTO `GRUPOS` VALUES ( 'NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' , '
NETCOM' , 'NETCOM' , 'NETCOM' ) ;
/_!40000 ALTER TABLE `GRUPOS` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` home dpto `
□□
DROP TABLE IF EXISTS `HOME DPTO` ;
CREATE TABLE `HOME DPTO` (
`URL Imagen1 ` varchar (50) d e f a u l t NULL,
`URL Imagen2 ` varchar (50) d e f a u l t NULL,
`URL Imagen3 ` varchar (50) d e f a u l t NULL,
` Di r e c c ion ` varchar (50) NOT NULL d e f a u l t '' ,
` Telefono Dpto ` varchar (20) NOT NULL d e f a u l t '' ,
` Telefono Uni ` varchar (20) d e f a u l t NULL,
`Fax ` varchar (50) d e f a u l t NULL,
` E Mai l ` varchar (50) NOT NULL d e f a u l t '' ,
`Nombre ` varchar (100) NOT NULL d e f a u l t '' ,
` Accesos ` varchar (1000) ,
`Mapas ` varchar (1000) ,
`Acomodacion ` varchar (1000) ,
PRIMARY KEY ( ` Telefono Dpto ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` home dpto `
□□
/_!40000 ALTER TABLE `HOME DPTO` DISABLE KEYS _/;
INSERT INTO `HOME DPTO` VALUES ( ' . / images / logo . png ' , ' . / images / l emai t . g i f ' , ' . / images / cuadros .
jpg ' , 'Avda . Univer s idad , 30 , Edi f . Tor res Quevedo . E□289 ' , '(+34) 91□7452344 ' , '(+34)
91□4562345 ' , '(+34) 91□624□8749 ' , ' s e c r e@i t . uc3m. es ' , ' Departamento de I n g e n i e r__ a Telem_at ica □ Unive r s idad Car
los I I I de Madrid ' ,NULL, '<i f rame width="640" he ight="480" f rameborder="0"
s c r o l l i n g="no" marginheight="0" marginwidth="0" s r c="ht tp : / /maps . googl e . e s /maps/ms? i e=UTF8&
amp ;msa=0&amp ; msid=104559785229237204753.00043 ac94c2c52fb9e658&amp ;om=1&amp ; s=
AARTsJopBo30scmGkek2plgh0G2Ttx7Q1w&amp ; l l =40.335684 ,□3.765264&amp ; spn=0.015702 ,0.027466&amp ;
z=15&amp ; iwl o c=00043 ac969ea7883d32d5&amp ; output=embed"></i f rame><br /><smal l><a h r e f="ht tp
: / /maps . googl e . e s /maps/ms? i e=UTF8&amp ;msa=0&amp ; msid=104559785229237204753.00043
ac94c2c52fb9e658&amp ;om=1&amp ; l l =40.335684 ,□3.765264&amp ; spn=0.015702 ,0.027466&amp ; z=15&amp ;
iwl o c=00043 ac969ea7883d32d5&amp ; sour c e=embed" s t y l e="c o l o r :#0000FF; text□a l i g n : l e f t " shape="
r e c t">Ver mapa m_as grande</a></smal l >' , 'Acomodacion ' ) ;
/_!40000 ALTER TABLE `HOME DPTO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` imparte `
□□
DROP TABLE IF EXISTS `IMPARTE` ;
CREATE TABLE `IMPARTE` (
`EmpleadoID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t '0 ' ,
` CodigoAsig ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t '0 ' ,
PRIMARY KEY ( ` EmpleadoID ` , ` CodigoAsig ` ) ,
KEY ` FK imparte 2 ` ( ` CodigoAsig ` ) ,
CONSTRAINT ` FK imparte 1 ` FOREIGN KEY ( ` EmpleadoID ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK imparte 2 ` FOREIGN KEY ( ` CodigoAsig ` ) REFERENCES `ASIGNATURAS` ( ` Codigo ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` EmpleadoID ` ) REFER `
webdpto/ empleados ' ;
```

```
□□
□□ Dumping data f o r t abl e `webdpto ` . ` imparte `
□□
/_!40000 ALTER TABLE `IMPARTE` DISABLE KEYS _/;
INSERT INTO `IMPARTE` VALUES (8 ,2334) ;
INSERT INTO `IMPARTE` VALUES (10 ,2334) ;
INSERT INTO `IMPARTE` VALUES (11 ,2334) ;
INSERT INTO `IMPARTE` VALUES (1 ,34233) ;
INSERT INTO `IMPARTE` VALUES (2 ,34233) ;
INSERT INTO `IMPARTE` VALUES (4 ,34233) ;
/_!40000 ALTER TABLE `IMPARTE` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` l i n e a s i n v e s t i g a c i o n `
□□
DROP TABLE IF EXISTS `LINEAS INVESTIGACION ` ;
CREATE TABLE `LINEAS INVESTIGACION` (
` Ti tulo ` varchar (50) NOT NULL d e f a u l t '' ,
` Ti t l e ` varchar (50) d e f a u l t NULL,
` Ti tulo Abreviado ` varchar (10) NOT NULL d e f a u l t '' ,
` Abbr eviat ed Ti t l e ` varchar (10) d e f a u l t NULL,
` De s c r ipc ion ` varchar (1000) NOT NULL,
` De s c r ipt ion ` varchar (1000) ,
`Grupo ` varchar (10) NOT NULL d e f a u l t '' ,
`URL Home Es ` varchar (50) d e f a u l t NULL,
`URL Home En ` varchar (50) d e f a u l t NULL,
PRIMARY KEY ( ` Ti tulo Abr eviado ` ) ,
KEY ` FK l ineas 1 ` ( `Grupo ` ) ,
CONSTRAINT ` FK l ineas 1 ` FOREIGN KEY ( `Grupo ` ) REFERENCES `GRUPOS` ( ` Ti tulo Abr eviado ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( `Grupo ` ) REFER `webdpto/
grupos ` ( ` Ti tul ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` l i n e a s i n v e s t i g a c i o n `
□□
/_!40000 ALTER TABLE `LINEAS INVESTIGACION` DISABLE KEYS _/;
INSERT INTO `LINEAS INVESTIGACION` VALUES ( 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' , 'GAST' ,
NULL,NULL) ;
INSERT INTO `LINEAS INVESTIGACION` VALUES ( ' l i n e a Gast ' , ' l i n e a Gast ' , ' l i n e a Gast ' , ' l i n e a Gast
' , ' l i n e a Gast ' , ' l i n e a Gast ' , 'GAST' , ' l i n e a Gast ' , ' l i n e a Gast ' ) ;
INSERT INTO `LINEAS INVESTIGACION` VALUES ( 'NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' , '
NETCOM' , 'NETCOM' , 'NETCOM' , 'NETCOM' ) ;
/_!40000 ALTER TABLE `LINEAS INVESTIGACION` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` ocupa `
□□
DROP TABLE IF EXISTS `OCUPA` ;
CREATE TABLE `OCUPA` (
`EmpleadoID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
`DespachoID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
PRIMARY KEY ( ` EmpleadoID ` , ` DespachoID ` ) ,
KEY ` FK ocupa 2 ` ( ` DespachoID ` ) ,
CONSTRAINT ` FK ocupa 1 ` FOREIGN KEY ( ` EmpleadoID ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK ocupa 2 ` FOREIGN KEY ( ` DespachoID ` ) REFERENCES `DESPACHO` ( ` DespachoID ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` ocupa `
□□
/_!40000 ALTER TABLE `OCUPA` DISABLE KEYS _/;
INSERT INTO `OCUPA` VALUES ( 1 , 1 ) ;
INSERT INTO `OCUPA` VALUES ( 2 , 1 ) ;
INSERT INTO `OCUPA` VALUES ( 8 , 1 ) ;
INSERT INTO `OCUPA` VALUES ( 1 1 , 1 ) ;
INSERT INTO `OCUPA` VALUES ( 2 , 2 ) ;
INSERT INTO `OCUPA` VALUES ( 3 , 2 ) ;
INSERT INTO `OCUPA` VALUES ( 4 , 2 ) ;
INSERT INTO `OCUPA` VALUES ( 9 , 2 ) ;
INSERT INTO `OCUPA` VALUES ( 1 0 , 3 ) ;
/_!40000 ALTER TABLE `OCUPA` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` pa r t i c ipa en `
□□
DROP TABLE IF EXISTS `PARTICIPA EN` ;
CREATE TABLE `PARTICIPA EN` (
` ProyectoID ` varchar (20) NOT NULL d e f a u l t '' ,
` LineaID ` varchar (10) NOT NULL d e f a u l t '' ,
PRIMARY KEY ( ` ProyectoID ` , ` LineaID ` ) ,
KEY ` FK par t i c ipa en 2 ` ( ` LineaID ` ) ,
CONSTRAINT ` FK par t i c ipa en 1 ` FOREIGN KEY ( ` ProyectoID ` ) REFERENCES `PROYECTOS INVESTIGACION`
( ` ProyectoID ` ) ON UPDATE CASCADE,
CONSTRAINT ` FK par t i c ipa en 2 ` FOREIGN KEY ( ` LineaID ` ) REFERENCES `LINEAS INVESTIGACION` ( `
```

```
Ti tulo Abr eviado ` ) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` ProyectoID ` ) REFER `
webdpto/ proye c tos ' ;

□□
□□ Dumping data f o r t abl e `webdpto ` . ` pa r t i c ipa en `
□□
/_!40000 ALTER TABLE `PARTICIPA EN` DISABLE KEYS _/;
INSERT INTO `PARTICIPA EN` VALUES ( ' 3 4 2 3 4 2 3 ', 'GAST' ) ;
INSERT INTO `PARTICIPA EN` VALUES ( ' 5 6 5 6 ', 'NETCOM' ) ;
INSERT INTO `PARTICIPA EN` VALUES ( ' ISBP□34 ', 'NETCOM' ) ;
/_!40000 ALTER TABLE `PARTICIPA EN` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` pe r t ene c e a `
□□
DROP TABLE IF EXISTS `PERTENECE A` ;
CREATE TABLE `PERTENECE A` (
` CodigoAsig ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ',
` Ti tulac ionID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ',
PRIMARY KEY ( ` CodigoAsig ` , ` Ti tulacionID ` ) ,
KEY ` FK per tenece a 2 ` ( ` Ti tulac ionID ` ) ,
CONSTRAINT ` FK per tenece a 1 ` FOREIGN KEY ( ` CodigoAsig ` ) REFERENCES `ASIGNATURAS` ( ` Codigo ` )
ON UPDATE CASCADE,
CONSTRAINT ` FK per tenece a 2 ` FOREIGN KEY ( ` Ti tulac ionID ` ) REFERENCES `TITULACIONES` ( ` ID ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` CodigoAsig ` ) REFER `
webdpto/ as ignatur ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` pe r t ene c e a `
□□
/_!40000 ALTER TABLE `PERTENECE A` DISABLE KEYS _/;
INSERT INTO `PERTENECE A` VALUES (34233 ,7) ;
INSERT INTO `PERTENECE A` VALUES (34233 ,8) ;
/_!40000 ALTER TABLE `PERTENECE A` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` p r o y e c t o s i n v e s t i g a c i o n `
□□
DROP TABLE IF EXISTS `PROYECTOS INVESTIGACION` ;
CREATE TABLE `PROYECTOS INVESTIGACION` (
` ProyectoID ` varchar (20) NOT NULL d e f a u l t '',
` Ti tulo ` varchar (50) NOT NULL d e f a u l t '',
` Ti t l e ` varchar (50) d e f a u l t NULL,
` Ti tulo Abreviado ` varchar (10) NOT NULL d e f a u l t '',
` Abbr eviat ed Ti t l e ` varchar (10) d e f a u l t NULL,
`Tipo ` varchar (30) NOT NULL d e f a u l t '',
` Fe cha Ini c i o ` date NOT NULL d e f a u l t '0000□00□00 ',
` Fecha Final ` date NOT NULL d e f a u l t '0000□00□00 ',
` Categor ia ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t ' 0 ',
`URL Externo ` varchar (50) d e f a u l t NULL,
`URL Local Es ` varchar (50) d e f a u l t NULL,
`URL Local En ` varchar (50) d e f a u l t NULL,
`URL Logo ` varchar (50) d e f a u l t NULL,
` De s c r ipc ion ` varchar (1000) ,
` De s c r ipt ion ` varchar (1000) ,
` Coordinador ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ',
` Responsable ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ',
PRIMARY KEY ( ` ProyectoID ` ) ,
KEY ` FK proyectos 1 ` ( ` Categor ia ` ) ,
KEY ` FK proyectos 2 ` ( ` Coordinador ` ) ,
KEY ` FK proyectos 3 ` ( ` Responsable ` ) ,
CONSTRAINT ` FK proyectos 1 ` FOREIGN KEY ( ` Categor ia ` ) REFERENCES `TIPOS PROYECTO` ( ` ID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK proyectos 2 ` FOREIGN KEY ( ` Coordinador ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` )
ON UPDATE CASCADE,
CONSTRAINT ` FK proyectos 3 ` FOREIGN KEY ( ` Responsable ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` )
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` Categor ia ` ) REFER `webdpto
/ t ipo s pr oy ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` p r o y e c t o s i n v e s t i g a c i o n `
□□
/_!40000 ALTER TABLE `PROYECTOS INVESTIGACION` DISABLE KEYS _/;
INSERT INTO `PROYECTOS INVESTIGACION` VALUES ( ' 3 4 2 3 4 2 3 ', ' p222 ', ' p222 ', ' SIP ', ' SIP1 ', ' p222
', '2008□03□31 ', '2008□03□31 ',2 , ' p222 ', ' p222 ', ' p222 ', '', ' p222 ', ' p222 ', 4 , 4 ) ;
INSERT INTO `PROYECTOS INVESTIGACION` VALUES ( ' 5 6 5 6 ', ' proyecto2 ', ' proyecto2 ', 'SAP', ' SAP1 ', '
proyecto2 ', '2008□03□22 ', '2008□03□22 ',2 , ' proyecto2 ', ' proyecto2 ', ' proyecto2 ', ' zvczxvcxv ', '
proyecto2 ', ' proyecto2 ', 2 , 2 ) ;
INSERT INTO `PROYECTOS INVESTIGACION` VALUES ( ' ISBP□34 ', ' Te cnologi e s ', ' Tecnology ', ' IS ', ' IS ', '
Te cnologi e s ', '2008□06□05 ', '2011□06□05 ',2 , ' sdas ', ' asdas ', ' asdsa ', '', ' asdasd ', ' asdasd ', 1 , 8 ) ;
/_!40000 ALTER TABLE `PROYECTOS INVESTIGACION` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` publ i c a c i one s `
□□
DROP TABLE IF EXISTS `PUBLICACIONES ` ;
CREATE TABLE `PUBLICACIONES` (
` Publ icacionID ` i n t (10) uns igned NOT NULL auto increment ,
```

```
` Ti tulo ` varchar (100) NOT NULL d e f a u l t '' ,
`Keywords ` varchar (50) d e f a u l t NULL,
`Anio ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t '0 ',
`EntidadXML ` blob NOT NULL,
` Responsable ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t '0 ',
`URL Externa ` varchar (50) d e f a u l t NULL,
PRIMARY KEY ( ` Publ icacionID ` ) ,
KEY ` FK publ i cac ione s 1 ` ( ` Responsable ` ) ,
CONSTRAINT ` FK publ i cac ione s 1 ` FOREIGN KEY ( ` Responsable ` ) REFERENCES `EMPLEADOS` ( `
EmpleadoID ` ) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` Responsable ` ) REFER `
webdpto/empleado ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` publ i c a c i one s `
□□
/_!40000 ALTER TABLE `PUBLICACIONES` DISABLE KEYS _/;
INSERT INTO `PUBLICACIONES` VALUES ( 1 , ' publ i c a c i on ' , ' new , s a l s a , zanahor ia ' , 2 0 0 8 , 0
x7A64666473667364 , 2 , ' ds f sdf ' ) ;
INSERT INTO `PUBLICACIONES` VALUES ( 2 , ' pubic23 ' , ' pubic23 ' , 2 0 0 8 , 0 x70756269633233 , 1 , ' pubic23 ' ) ;
INSERT INTO `PUBLICACIONES` VALUES ( 3 , ' fdf ' , ' f sdf ' , 3 2 4 , 0 x617364617364 , 3 , ' asdas ' ) ;
/_!40000 ALTER TABLE `PUBLICACIONES` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t e s i s `
□□
DROP TABLE IF EXISTS `TESIS ` ;
CREATE TABLE `TESIS ` (
` TesisID ` i n t (10) uns igned NOT NULL auto increment ,
` Ti tulo ` varchar (100) NOT NULL d e f a u l t '' ,
` Ti t l e ` varchar (100) d e f a u l t NULL,
`Autor ` varchar (50) NOT NULL d e f a u l t '' ,
` Autor Local ` sma l l i n t ( 5 ) uns igned d e f a u l t NULL,
` Fecha ` date d e f a u l t NULL,
`Resumen ` varchar (1000) ,
` Abstract ` varchar (1000) ,
`URL PDF` varchar (50) d e f a u l t NULL,
` Di r e c tor ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t '0 ',
` Codi r e c tor ` varchar (50) d e f a u l t NULL,
`Doctorado ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0 ',
PRIMARY KEY ( ` TesisID ` ) ,
KEY ` FK t e s i s 1 ` ( ` Di r e c tor ` ) ,
KEY ` FK t e s i s 2 ` ( ` Autor Local ` ) ,
CONSTRAINT ` FK t e s i s 1 ` FOREIGN KEY ( ` Di r e c tor ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` ) ON
UPDATE CASCADE,
CONSTRAINT ` FK t e s i s 2 ` FOREIGN KEY ( ` Autor Local ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` ) ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` Autor Local ` ) REFER `
webdpto/empleado ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t e s i s `
□□
/_!40000 ALTER TABLE `TESIS ` DISABLE KEYS _/;
INSERT INTO `TESIS ` VALUES ( 1 , ' Te s i s e j emplo ' ,NULL, ' Autor de Tes i s ' , 1 ,NULL,NULL,NULL,NULL, 3 ,
NULL, 0 ) ;
INSERT INTO `TESIS ` VALUES ( 2 , ' t e s i s ' , ' t e s i s ' , ' t e s i s ' ,2 , '2008 □03 □22 ' , ' t e s i s ' , ' t e s i s ' , ' t e s i s
' , 2 , ' t e s i s ' , 1 ) ;
/_!40000 ALTER TABLE `TESIS ` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t i po s c a r g o `
□□
DROP TABLE IF EXISTS `TIPOS CARGO` ;
CREATE TABLE `TIPOS CARGO` (
` ID ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0 ',
`Cargo ` varchar (30) NOT NULL d e f a u l t '' ,
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t i po s c a r g o `
□□
/_!40000 ALTER TABLE `TIPOS CARGO` DISABLE KEYS _/;
INSERT INTO `TIPOS CARGO` VALUES ( 1 , ' d i r e c t o r dpt o ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 2 , ' r e spons abl e a r e a ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 3 , ' s e c r e t a r i o dpt o ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 4 , ' subdi r e c t o r do c enc i a ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 5 , ' s u b d i r e c t o r l a b o r a t o r i o s ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 6 , ' s u b d i r e c t o r g e n e r a l ' ) ;
INSERT INTO `TIPOS CARGO` VALUES ( 7 , ' coordinador e rasmus ' ) ;
/_!40000 ALTER TABLE `TIPOS CARGO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t i p o s c o n t r a t o `
□□
DROP TABLE IF EXISTS `TIPOS CONTRATO` ;
CREATE TABLE `TIPOS CONTRATO` (
```

```
` ID ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0',
` Contrato ` varchar (30) NOT NULL d e f a u l t '',
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;

□□
□□ Dumping data f o r t abl e `webdpto ` . ` t i p o s c o n t r a t o `
□□
/_!40000 ALTER TABLE `TIPOS CONTRATO` DISABLE KEYS _/;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 , ' ayudante ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 2 , ' asociado ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 3 , ' inve s t igador PDI ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 4 , ' ayudante doctor ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 5 , ' cont ratado doc tor ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 6 , ' v i s i t a n t e ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 7 , ' t i t u l a r i n t e r i n o ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 8 , ' t i t u l a r ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 9 , ' c a t edr _a t i c o ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 0 , ' t_ecnico PAS ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 1 , ' admin PAS ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 2 , ' becario PAS ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 3 , ' t_e cni c o pr o y e c t o ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 4 , ' inv e s t i g ado r pr o y e c t o ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 5 , ' admin proyecto ' ) ;
INSERT INTO `TIPOS CONTRATO` VALUES ( 1 6 , ' be car io dpto ' ) ;
/_!40000 ALTER TABLE `TIPOS CONTRATO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t ipo s ev ent o `
□□
DROP TABLE IF EXISTS `TIPOS EVENTO` ;
CREATE TABLE `TIPOS EVENTO` (
` ID ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0',
`Nombre ` varchar (50) NOT NULL d e f a u l t '',
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t ipo s e vent o `
□□
/_!40000 ALTER TABLE `TIPOS EVENTO` DISABLE KEYS _/;
INSERT INTO `TIPOS EVENTO` VALUES ( 1 , ' home enlaces ' ) ;
INSERT INTO `TIPOS EVENTO` VALUES ( 2 , ' home destacamos ' ) ;
INSERT INTO `TIPOS EVENTO` VALUES ( 3 , ' home eventos ' ) ;
INSERT INTO `TIPOS EVENTO` VALUES ( 4 , ' home ' ) ;
INSERT INTO `TIPOS EVENTO` VALUES ( 5 , ' eventos ' ) ;
/_!40000 ALTER TABLE `TIPOS EVENTO` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t i p o s p e r f i l `
□□
DROP TABLE IF EXISTS `TIPOS PERFIL ` ;
CREATE TABLE `TIPOS PERFIL ` (
` ID ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0',
` Pe r f i l ` varchar (30) NOT NULL d e f a u l t '',
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t i p o s p e r f i l `
□□
/_!40000 ALTER TABLE `TIPOS PERFIL ` DISABLE KEYS _/;
INSERT INTO `TIPOS PERFIL ` VALUES ( 1 , ' admini s t rador ' ) ;
INSERT INTO `TIPOS PERFIL ` VALUES ( 2 , ' admini s t r a t ivo ' ) ;
INSERT INTO `TIPOS PERFIL ` VALUES ( 3 , ' t_e cni c o ' ) ;
INSERT INTO `TIPOS PERFIL ` VALUES ( 4 , ' empleado dpto ' ) ;
/_!40000 ALTER TABLE `TIPOS PERFIL ` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t ipo s pr o y e c t o `
□□
DROP TABLE IF EXISTS `TIPOS PROYECTO` ;
CREATE TABLE `TIPOS PROYECTO` (
` ID ` t i n y i n t ( 3 ) uns igned NOT NULL d e f a u l t '0',
` Proyecto ` varchar (30) NOT NULL d e f a u l t '',
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t ipo s pr o y e c t o `
□□
/_!40000 ALTER TABLE `TIPOS PROYECTO` DISABLE KEYS _/;
INSERT INTO `TIPOS PROYECTO` VALUES ( 1 , ' i n t e r n a c i o n a l ' ) ;
INSERT INTO `TIPOS PROYECTO` VALUES ( 2 , ' nac ional ' ) ;
INSERT INTO `TIPOS PROYECTO` VALUES ( 3 , ' r e g i ona l ' ) ;
INSERT INTO `TIPOS PROYECTO` VALUES ( 4 , ' compa~n__a ' ) ;
/_!40000 ALTER TABLE `TIPOS PROYECTO` ENABLE KEYS _/;
```

```
□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t i t u l a c i o n e s `
□□
DROP TABLE IF EXISTS `TITULACIONES ` ;
CREATE TABLE `TITULACIONES` (
` ID ` sma l l i n t ( 5 ) uns igned NOT NULL auto increment ,
` Ti tulo ` varchar (50) NOT NULL d e f a u l t '' ,
` Ti t l e ` varchar (50) d e f a u l t NULL,
`URL Es ` varchar (50) NOT NULL d e f a u l t '' ,
`URL En ` varchar (50) d e f a u l t NULL,
PRIMARY KEY ( ` ID ` )
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t i t u l a c i o n e s `
□□
/_!40000 ALTER TABLE `TITULACIONES` DISABLE KEYS _/;
INSERT INTO `TITULACIONES` VALUES ( 1 , ' I n g e n i e r i a de Telecomunicaciones ' , ' Telecommunication
Engineer ing ' , ' ht tp : / /www. uc3m. e s /uc3m/ g r a l /ES/ESCU/ e s cui 0 7 . html ' ,NULL) ;
INSERT INTO `TITULACIONES` VALUES ( 2 , ' I n g e n i e r i a I n d u s t r i a l ' ,NULL, ' ht tp : / /www. uc3m. e s /uc3m/
g r a l /ES/ESCU/ e s cui 0 7 . html ' ,NULL) ;
INSERT INTO `TITULACIONES` VALUES ( 7 , ' s i s t emas ' , ' s i s t emas ' , ' s i s t emas ' , ' s i s t emas ' ) ;
INSERT INTO `TITULACIONES` VALUES ( 8 , ' t e l emat i ca ' , ' t e l emat i ca ' , ' t e l emat i ca ' , ' t e l emat i ca ' ) ;
/_!40000 ALTER TABLE `TITULACIONES` ENABLE KEYS _/;


□□
□□ Table s t r u c t u r e f o r t abl e `webdpto ` . ` t rabaja en `
□□
DROP TABLE IF EXISTS `TRABAJA EN` ;
CREATE TABLE `TRABAJA EN` (
`EmpleadoID ` sma l l i n t ( 5 ) uns igned NOT NULL d e f a u l t ' 0 ' ,
` ProyectoID ` varchar (20) NOT NULL d e f a u l t '' ,
PRIMARY KEY ( ` EmpleadoID ` , ` ProyectoID ` ) ,
KEY ` FK t rabaja en 2 ` ( ` ProyectoID ` ) ,
CONSTRAINT ` FK t rabaja en 1 ` FOREIGN KEY ( ` EmpleadoID ` ) REFERENCES `EMPLEADOS` ( ` EmpleadoID ` )
ON UPDATE CASCADE,
CONSTRAINT ` FK t rabaja en 2 ` FOREIGN KEY ( ` ProyectoID ` ) REFERENCES `PROYECTOS INVESTIGACION`
( ` ProyectoID ` ) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=ut f 8 COMMENT='InnoDB f r e e : 3072 kB; ( ` EmpleadoID ` ) REFER `
webdpto/ empleados ' ;


□□
□□ Dumping data f o r t abl e `webdpto ` . ` t rabaja en `
□□
/_!40000 ALTER TABLE `TRABAJA EN` DISABLE KEYS _/;
INSERT INTO `TRABAJA EN` VALUES ( 1 , ' 3 4 2 3 4 2 3 ' ) ;
INSERT INTO `TRABAJA EN` VALUES ( 2 , ' 3 4 2 3 4 2 3 ' ) ;
INSERT INTO `TRABAJA EN` VALUES ( 4 , ' 3 4 2 3 4 2 3 ' ) ;
INSERT INTO `TRABAJA EN` VALUES ( 2 , ' 5 6 5 6 ' ) ;
INSERT INTO `TRABAJA EN` VALUES ( 3 , ' 5 6 5 6 ' ) ;
INSERT INTO `TRABAJA EN` VALUES ( 4 , ' 5 6 5 6 ' ) ;
/_!40000 ALTER TABLE `TRABAJA EN` ENABLE KEYS _/;
/_!40101 SET SQL MODE=@OLD SQL MODE _/;
/_!40014 SET FOREIGN KEY CHECKS=@OLD FOREIGN KEY CHECKS _/;
/_!40014 SET UNIQUE CHECKS=@OLD UNIQUE CHECKS _/;
/_!40101 SET CHARACTER SET CLIENT=@OLD CHARACTER SET CLIENT _/;
/_!40101 SET CHARACTER SET RESULTS=@OLD CHARACTER SET RESULTS _/;
/_!40101 SET COLLATION CONNECTION=@OLD COLLATION CONNECTION _/;
/_!40101 SET CHARACTER SET CLIENT=@OLD CHARACTER SET CLIENT _/;


□□
□□ F i l t r o s para l a s t a bl a s de s cargadas (dump)
□□
□□
□□ F i l t r o s para l a t abl a `ANUNCIA GRUPO`
□□ ALTER TABLE `ANUNCIA GRUPO`
ADD CONSTRAINT ` anunc i a g rupo ibfk 2 ` FOREIGN KEY ( `GrupoID ` ) REFERENCES `GRUPOS` ( `
Ti tulo Abr eviado ` ) ,
ADD CONSTRAINT ` anunc i a g rupo ibfk 3 ` FOREIGN KEY ( ` EventoID ` ) REFERENCES `EVENTOS` ( ` EventoID
` ) ON UPDATE CASCADE;


□□
□□ F i l t r o s para l a t abl a `ANUNCIA PROYECTO`
□□ ALTER TABLE `ANUNCIA PROYECTO`
ADD CONSTRAINT ` anunc i a pr o y e c t o ibf k 5 ` FOREIGN KEY ( ` ProyectoID ` ) REFERENCES `
PROYECTOS INVESTIGACION` ( ` ProyectoID ` ) ON UPDATE CASCADE,
ADD CONSTRAINT ` anunc i a pr o y e c t o ibf k 6 ` FOREIGN KEY ( ` EventoID ` ) REFERENCES `EVENTOS` ( `
EventoID ` ) ON UPDATE CASCADE;


□□
□□ F i l t r o s para l a t abl a `ASIGNATURAS`
□□ ALTER TABLE `ASIGNATURAS`
ADD CONSTRAINT ` a s i g n a t u r a s i b f k 1 ` FOREIGN KEY ( ` Coordinador ` ) REFERENCES `EMPLEADOS` ( `
EmpleadoID ` ) ON UPDATE CASCADE;
```

```
--
-- Filtros para la tabla `DESPACHO`
-- ALTER TABLE `DESPACHO`
ADD CONSTRAINT `despachoibfk1` FOREIGN KEY (`Campus`) REFERENCES `CAMPUS` (`CampusID`);


--
-- Filtros para la tabla `EMPLEADOS`
-- ALTER TABLE `EMPLEADOS`
ADD CONSTRAINT `empleadosibfk2` FOREIGN KEY (`Contrato`) REFERENCES `TIPOS CONTRATO` (`ID`),
ADD CONSTRAINT `empleadosibfk3` FOREIGN KEY (`Perfil`) REFERENCES `TIPOS PERFIL` (`ID`),
ADD CONSTRAINT `empleadosibfk4` FOREIGN KEY (`Cargo`) REFERENCES `TIPOS CARGO` (`ID`),
ADD CONSTRAINT `empleadosibfk5` FOREIGN KEY (`Grupo`) REFERENCES `GRUPOS` (`Titulo Abreviado
`);


--
-- Filtros para la tabla `ES AUTOR`
-- ALTER TABLE `ES AUTOR`
ADD CONSTRAINT `esautoribfk1` FOREIGN KEY (`EmpleadoID`) REFERENCES `EMPLEADOS` (`
EmpleadoID`),
ADD CONSTRAINT `esautoribfk2` FOREIGN KEY (`PublicacionID`) REFERENCES `PUBLICACIONES` (`
Publ
```

10.2 Anéxo B.

10.2.1 Casos de Uso Extendidos: Templates.

En el siguiente anéxo se presentan todos los casos de uso ideados para el funcionamiento de la aplicación. Para cada uno de ellos se muestra los perfiles autorizados a realizarlos, permisos, flujo de datos, precondiciones, postcondiciones, casos de uso subordinados, posibles excepciones, etc. Cabe decir que el caso de uso "Login" es un caso de uso subordinado de todos los demás (cualquier operación en la aplicación requiere que el usuario este identificado y autorizado) pero que no será indicado en los templates para evitar repetición.

En esta versión de la aplicación no están disponibles todas las funcionalidades aquí descritas ya que estas se han ido añadiendo a medida que avanzabamos en el desarrollo del proyecto o mejorandolas al ver los resultados del prototipo creado. Para hacer notar esto al lector, marcaremos en cursiva todas aquellas funcionalidades que no están disponibles o que aún no están implementadas de la forma descrita. En el capítulo 4.6 de esta memoria, el lector tiene un resumen de los casos de uso implementados con su funcionamiento además de una pequeña comparación con los casos de uso a continuación descritos.

En cuanto a los actores que intervienen en cada uno de los casos de uso, estos representan los perfiles dentro de la aplicación. Según el sistema de perfiles dispuesto, tenemos cuatro tipos de actores: Administrador, Personal Administrativo, Empleado, Técnico, enumerados de mayor a menor según los permisos que estos tienen sobre la aplicación. Además, según está implementada la aplicación en estos momentos, el perfil con mayores permisos es el que tiene prioridad, es decir, un empleado con perfil de administrador siempre entrará como administrador en la aplicación no como empleado. Esto se sobreentiende en esta versión de la aplicación pero para posteriores implementaciones se ha pensado en usuarios con múltiples perfiles por lo que hace falta reseñar este funcionamiento.

Caso de Uso	LOGIN
Descripción	El usuario intenta acceder al sistema y para ello debe logarse a través del servidor *LDAPS* (del departamento o de uno de los grupos de investigación del departamento). La primera versión utilizará LDAP en vez de LDAPS.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto Personal Administrativo Técnico
Precondiciones	El usuario debe estar dado de alta en la BBDD *LDAPS*
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca su nombre de usuario y su password. 2. El usuario introduce los datos. 3. El sistema valida estos datos a través de *LDAPS* 4. Si son correctos: 1. El usuario entra en el sistema. 2. *Si logging está activado, el sistema añade una anotación al registro histórico.* 5. Si no son correctos: a) El sistema vuelve a pedir al usuario que introduzca su nombre de usuario y su password. b) *Después de tres intentos fallidos, el sistema bloquea el acceso desde el IP del usuario durante un tiempo aleatorio, **Notificar Interesados**, y, si logging está activado, añade una anotación al registro histórico.*
Postcondiciones	1. El usuario tendrá acceso restringido a determinadas áreas y operaciones dependiendo de su perfil.
Excepciones	1. Error de autenticación (el usuario no está en BBDD LDAPS) 2. Error interno (ej. conexión a BBDD)
Casos de uso subordinados	***Notificar Interesados** donde los interesados son:* • *el administrador.* *(solo en el caso de 3 intentos de login fallidos)*

10.2.1.1 Gestión de empleados

Caso de Uso	CREAR EMPLEADO
Descripción	El usuario da de alta a un nuevo empleado.
Prioridad	1 (Primera fase)
Actores	Personal Administrativo Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1A. Si el usuario tiene el perfil de Administrador a) El sistema le pide que introduzca todos los datos del nuevo empleado. 1B. Si el usuario tiene el perfil de Personal Administrativo b) El sistema le pide que introduzca todos los datos del nuevo empleado menos los "datos de cuenta"[2]. 2. El usuario introduce los datos, en particular: a) El usuario puede eligir un grupo de investigación para el nuevo empleado (**Asignar Grupo**). b) El usuario elige al menos un despacho para el nuevo empleado (**Designar Despachos**). 3. El usuario cancela o acepta la operación a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor salvo el login_local), excepción. b) Si se acepta sin haber puesto valor al atributo login_local, el sistema genera un valor único provisional (ej. el valor del atributo EmpleadoID). 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El empleado debe tener un primerNombre, un primerApellido, un login_local, un despacho, un contrato, una fecha_inicio, un perfil y una indicación de si es doctor o no. 2. Un empleado que no sea: a) un PDI con contrato de un asociado, b) un empleado de docencia (del departamento), c) un PAS debe ser miembro de un grupo de investigación. 3. El e-mail (si existe), el login_local y la URL_pagina (si existe) del empleado deben ser únicos.

[2] Los "datos de cuenta" de un empleado se definen actualmente como los atributos siguientes: login_local, e-mail y URL_pagina.

Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error cometer (datos inválidos; volver al paso 1)
Casos de uso Subordinados	**Designar Despachos** **Asignar Grupo** (en su caso) *Nota: implica **Notificar Interesados*** (*responsable del grupo*). **Notificar Interesados**, donde los interesados son: • los técnicos, • el personal administrativo • el administrador

Caso de Uso	MODIFICAR EMPLEADO
Descripción	El usuario modifica los datos de un empleado.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (el propio empleado) Técnico Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El empleado que se quiere modificar debe estar dado de alta en el sistema. 4. Si el usuario tiene el perfil de Empleado Dpto sólo tiene acceso al empleado que le representa.
Flujo Básico de Datos	1. El sistema le pide que seleccione un empleado de entre los empleados a los que tiene acceso. 2. El usuario selecciona un empleado 3A. Si el perfil del usuario es Administrador a) el sistema le ofrece la posibilidad de modificar cualquier dato del empleado seleccionado. 3B. Si el perfil del usuario es Personal Administrativo a) el sistema le ofrece la posibilidad de modificar cualquier dato del empleado seleccionado menos los "datos de cuenta". 3C. Si el perfil del usuario es Técnico, a) el sistema le ofrece la posibilidad de modificar sólo los "datos de cuenta" del empleado seleccionado. 3D. Si el perfil del usuario es Empleado Dpto a) El sistema le ofrece la posibilidad de modificar solo sus "datos personales"[3] del empleado seleccionado. 4. El usuario introduce las modificaciones, en particular: a) El usuario puede modificar el grupo de investigación del empleado (**Asignar Grupo**) si tiene el perfil adecuado. b) El usuario puede modificar la lista de despachos que ocupa el empleado (**Designar Despachos**). 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. *b) Si el usuario es Técnico y se acepta con el valor del login_local puesto al valor provisional, excepción* 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una*

[3] Los "datos personales" de un empleado se definen actualmente como los atributos siguientes: Nombre primero y segundo, Apellidos primero y segundo, (posible: E-mail, URL_pagina), cambio de despacho.

	anotación al registro histórico. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El empleado debe tener un primerNombre, un primerApellido, un login_local, un despacho, un contrato, una fecha_inicio, un perfil y una indicación de si es doctor o no. 2. Un empleado que no sea a) un PDI con contrato de Asociado, b) un empleado de docencia (del departamento), c) un PAS debe ser miembro de un grupo de investigación. 3. El e-mail (si existe), el login_local y la URL_pagina (si existe) del empleado deben ser únicos. 4. Caso de un usuario con el perfil de Empleado Dpto: a) No puede haberse cambiado en la base de datos ningún dato de un empleado que no sea el usuario actual. b) No puede haberse cambiado en la base de datos ningún dato que no forme parte de los "datos personales" del empleado que es el usuario actual.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error cometer (datos inválidos; volver al paso 3)
Casos de uso subordinados	**Designar Despachos** **Asignar Grupo** (en su caso). *Nota: implica* ***Notificar Interesados*** (*responsable del grupo*). **Notificar Interesados** donde los interesados son: • el empleado • el personal administrativo • el administrador

Caso de Uso	BORRAR EMPLEADO
Descripción	El usuario elimina a un empleado. No hace falta borrar a los empleados que se han ido; no se mostrarán entre el personal actual de las páginas Web generadas si la fecha_final del contrato ya ha pasado.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El empleado que se quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un empleado de entre los empleados existentes. 2. El usuario selecciona a un empleado. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma: a) Si el empleado es coordinador de una asignatura, *un grupo* o un proyecto; es responsable de *un grupo*, un proyecto o una publicación; es director de una tesis; o tiene un cargo; excepción borrado 1. b) Si el empleado es es el único autor_local de alguna publicación, el único miembro de algún grupo, el unico trabajador de algún proyecto o el único profesor de alguna asignatura, excepción borrado 2. 6. El sistema elimina: a) el empleado b) el resto de las relaciones que tiene con otras entidades (proyectos, asignaturas, publicaciones, tesis). 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Un grupo de investigación no puede tener como miembro, *como responsable o como coordinador* a un empleado inexistente. 2. Una asignatura no puede tener como profesor o coordinador a un empleado inexistente. 3. Un proyecto de investigación no puede tener como trabajador oficial, coordinador (es decir, contacto) o responsable a un empleado inexistente. 4. Una publicación no puede tener como responsable o como autor a un empleado inexistente;

	5. Una tesis no puede tener como director o como autor_local a un empleado inexistente 6. Un despacho no puede estar ocupado por un empleado inexistente
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado 1 (el sistema comunica al usuario que tiene que reasignar las relaciones obligatorias que tiene este empleado con otras entidades antes de poder eliminarlo y ofrece facilidades para hacerlo; volver al paso 5b). 3. Error borrado 2 (el sistema desaconseja el borrado; si el usuario insiste: se borra tambien el grupo / la publicación / el proyecto / la asignatura en cuestion, en el último caso solo si el modelo de datos no permite asignaturas sin profesores como es el caso actualmente; volver al paso 6).
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el personal administrativo • el administrador

10.2.1.2 Gestión de asignaturas

En el modelo de datos actual, una asignatura tiene que pertenecer a al menos una titulación y tiene que haber al menos un profesor que la imparte. Probablemente habrá que quitar estas dos restricciones, sobre todo la última, para permitir guardar datos de asignaturas que todavía no se imparten o que ya no se imparten. Pero si se quitan estas restricciones, se tiene que definir los criterios para publicar o no una asignatura en las páginas Web (p.e. un campo 'boolean' "activa").

Del mismo modo, si se quiere tener titulaciones sin asignaturas o con sólo asignaturas que no se imparten, se tiene que definir los criterios para publicarlas o no en las páginas Web (otra vez, podría ser un campo 'boolean' "activa").

Caso de Uso	CREAR ASIGNATURA
Descripción	El usuario da de alta una nueva asignatura.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. Debe haber al menos un PDI doctor dado de alta. 4. Debe haber un empleado que tiene asignado el cargo de subdirector de docencia.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos de la nueva asignatura. 2. El usuario introduce los datos; en particular: a) El usuario elige a un coordinador de la asignatura de entre los doctores (**Asignar Empleado**); *si no se asigna uno, el sistema establecerá el empleado que tiene el cargo de subdirector de docencia como coordinador*. b) El usuario puede eligir a los profesores de entre los empleados PDI (**Designar Empleados**). c) El usuario elige al menos una titulación a la que pertenece la asignatura (**Designar Titulaciones**). 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La asignatura debe tener: un código, un título, un curso, un cuatrimestre, una URL_uni_es, un coordinador, al menos una titulación a la que pertenece y, en el modelo de datos actual, al menos un profesor que la imparte.

	2. El código, la URL_uni_es y la URL_uni_en (si existe) de la asignatura deben ser únicos. 3. El coordinador de una asignatura puede no ser uno de los profesores que imparta la asignatura pero siempre deberá ser PDI doctor. 4. Los profesores de una asignatura deben ser PDI (*podría relajarse esta restricción del modelo de datos y simplemente dar un aviso si se elige un profesor no PDI*).
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1). 3. Error de asignación (no existe el cargo de subdirector de docencia o existe pero ningún empleado lo tiene asignado; termina el caso de uso).
Casos de uso subordinados	**Designar Titulaciones** **Asignar Empleado** (en su caso) donde el empleado es: • el coordinador. Nota: implica **Notificar Interesados** (coordinador) **Designar Empleados** (en su caso) donde los empleados son: • los profesores. Nota: implica **Notificar Interesados** (profesores) **Notificar Interesados** donde los interesados son: • personal administrativo • el administrador.

Caso de Uso	MODIFICAR ASIGNATURA
Descripción	El usuario modifica los datos de una asignatura existente.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (coordinador de la asignatura) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La asignatura que se quiere modificar debe estar dada de alta en el sistema. 4. Si el perfil del usuario es Empleado Dpto, solo tiene acceso a las asignaturas de las que es coordinador.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una asignatura de entre las asignaturas a las que tiene acceso. 2. El usuario selecciona una asignatura. *3. Si el perfil del usuario es Empleado Dpto:* *a) El sistema le ofrece la posibilidad de modificar los "datos básicos"[4] de la asignatura seleccionada.* 4. El usuario introduce las modificaciones, en particular a) puede modificar el coordinador (**Asignar Empleado**) y/o los profesores (**Designar Empleados**) si tiene el perfil adecuado. b) El usuario puede modificar la lista de titulaciones a las que pertenece la asignatura (**Designar Titulaciones**). 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La asignatura debe tener: un código, un título, un curso, un cuatrimestre, una URL_uni_es, un coordinador al menos una titulación a la que pertenece y, en el modelo de datos actual, al menos un profesor que la imparte. 2. El código, la URL_uni_es y la URL_uni_en (si existe) de la asignatura deben ser únicos. 3. El coordinador de una asignatura puede no ser uno de los profesores que imparta la asignatura pero siempre deberá

[4] Los "datos básicos" de una asignatura se definen actualmente como los atributos siguientes: las cuatro URLs y el título (en inglés y en español).

	ser PDI doctor. 4. *Los profesores de una asignatura deben ser PDI* (*podría relajarse esta restricción del modelo de datos y simplemente dar un aviso si se elige un profesor no PDI*). 5. Caso de un usuario que tiene el perfil de Empleado Dpto: *a) No puede haberse cambiado en la base de datos los datos no básicos de ninguna asignatura.* b) No puede haberse cambiado en la base de datos ningún dato de una asignatura de la que el empleado que es el usuario actual no es coordinador.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3).
Casos de uso subordinados	**Designar Titulaciones** **Asignar Empleado** (en su caso) donde el empleado es: • el coordinador nuevo Nota: implica **Notificar Interesados** (coordinador nuevo y antiguo) **Designar Empleados** (en su caso) donde los empleados son: • los profesores añadidos y eliminados Nota: implica **Notificar Interesados**. (profesores añadidos y eliminados) **Notificar Interesados** donde los interesados son: • el coordinador • el administrador

Caso de Uso	BORRAR ASIGNATURA
Descripción	El usuario elimina una asignatura del sistema. Con el modelo de datos actual, se tiene que borrar una asignatura que ya no se da porque una asignatura tiene que pertenecer a al menos una titulación.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La asignatura que se quiere borrar debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una asignatura de entre las asignaturas existentes. 2. El usuario selecciona una asignatura. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado). 5. Si el usuario confirma, el sistema elimina a) la asignatura. b) las relaciones que tiene esta asignatura con otras entidades (empleados y titulaciones). 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Un empleado no puede estar impartiendo ni ser coordinador de una asignatura inexistente. 2. Una titulación no puede contener una asignatura inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el coordinador • los profesores • el personal administrativo • el administrador

Caso de Uso	CREAR TITULACIÓN
Descripción	El usuario crea una nueva titulación.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos de la nueva titulación. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. La titulación debe tener un título y un URL_es. 2. El título, el title, la URL_es y la URL_en de la titulación deben ser únicos.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

Caso de Uso	MODIFICAR TITULACIÓN
Descripción	El usuario quiere modificar los datos de una titulación existente.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La titulación que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una titulación de entre las titulaciones existentes. 2. El usuario elige una titulación. 3. El sistema ofrece al usuario la posibilidad de modificar los datos de la titulación. 4. El usuario introduce los datos. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. La titulación debe tener un título y un URL_es. 2. El título, el title, la URL_es y la URL_en de la titulación deben ser únicos.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)

Caso de Uso	BORRAR TITULACIÓN
Descripción	El usuario elimina una titulación del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La titulación que se quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una titulación de entre las titulaciones existentes. 2. El usuario elige una titulación. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado). 5. Si el usuario confirma a) Si la titulación es la única titulación a la que pertenece alguna asignatura, excepción. 6. El usuario elimina la titulación. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. Una asignatura no puede pertenecer a una titulación inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado (el sistema comunica al usuario que tiene que reasignar las asignaturas que no tienen otra titulación antes de poder borrarla y da facilidades para hacerlo; volver al paso 6).

10.2.1.3 Gestión de proyectos de investigación

En el modelo de datos actual, los proyectos se relacionan con los grupos de investigación a través de las líneas de investigación de los grupos (de interés para la generación de las páginas Web de los grupos).

Caso de Uso	CREAR PROYECTO
Descripción	El usuario da de alta a un nuevo proyecto. Se estudiará la posible integración con Universitas XXI para poder importar proyectos de esta o viceversa.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del proyecto. 2. El usuario introduce los datos, en particular: a) El usuario elige una o varias líneas de investigación de entre las líneas existentes para asignar el proyecto a ellas (**Designar Líneas**); si lo asigna a un grupo en vez de a una línea concreta, el proyecto se asocia a la línea de investigación por defecto de este grupo. b) El usuario elige a un coordinador (que también se considera trabajador) de entre los PDI, y posiblemente a un responsable (de los datos publicados en Web) (**Asignar Empleado**); *si no asigna ningún responsable, el sistema establecerá al propio usuario como responsable*. También puede elegir a más trabajadores (**Designar Empleados**). 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El proyecto debe tener: un título, un título_abreviado, un tipo (Esprit, etc.), una fecha_inicio, una fecha_final y una categoría (internacional, nacional, regional, compañía), un coordinador y un responsable (de los datos publicados en Web). 2. El proyecto debe tener al menos una línea de investigación que participa en el (puede ser la línea por defecto de uno de

	los grupos de investigación). 3. *El proyecto debe tener al menos un empleado que trabaja en el (satisfecho si tiene coordinador).* 4. El coordinador del proyecto debe ser PDI. 5. El título, el title (si existe), el título_abreviado, el abbreviated_title (si existe) la URL_externa (si existe) y la URL_logo (si existe) del proyecto deben ser únicos.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	**Designar Líneas** **Asignar Empleado** donde los empleados son: • el coordinador del proyecto, • el responsable del proyecto (por defecto, el usuario). Nota: implica **Notificar Interesados** (coordinador y responsable). **Designar Empleados** (en su caso) donde los empleados son: • los empleados que trabajan en el proyecto Nota: implica **Notificar Interesados** (trabajadores) **Notificar Interesados** donde los interesados son: • el administrador.

Caso de Uso	MODIFICAR PROYECTO
Descripción	El usuario quiere modificar los datos de un proyecto dado de alta en el sistema.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (coordinador o responsable) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El proyecto que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un proyecto de entre los proyectos existentes. 2. El usuario selecciona un proyecto 3. Si el usuario tiene el perfil de Empleado Dpto a) Si no es ni coordinador ni responsable del proyecto seleccionado, excepción (no se restringe con una precondición porque se quiere utilizar la excepción para comunicar el nombre del actual responsable, puesto que esta información no se publicará en las páginas Web). 4. El sistema le ofrece la posibilidad de modificar los datos del proyecto seleccionado 5. El usuario introduce las modificaciones; en particular: a) El usuario puede modificar el responsable y el coordinador (**Asignar Empleado**) y los trabajadores (**Designar Empleados**). b) El usuario puede modificar las líneas de investigación que participan en el proyecto (**Asignar Lineas**); si, al borrar la asociación entre un proyecto y una línea de investigación, el proyecto queda sin estar asociado a ninguna línea, se asocia a la línea por defecto del mismo grupo de la línea borrada. 6. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. El proyecto debe tener: un título, un título_abreviado, un tipo (Esprit, etc.), una fecha_inicio, una fecha_final y una categoría (internacional, nacional, autonomía, compañía), un coordinador, un responsable (de los datos publicados en Web) y al menos una línea de investigación que participa en

	el (puede ser la línea por defecto de uno de los grupos de investigación). También debe tener al menos un empleado que trabaja en el (satisfecho si el coordinador también es trabajador). 2. El título, el title (si existe), el título_abreviado, el abbreviated_title (si existe) la URL_externa (si existe) y la URL_logo (si existe) del proyecto deben ser únicos. 3. Caso de un usuario con el perfil de Empleado Dpto: a) No puede haberse cambiado en la base de datos ningún dato de un proyecto del que el empleado que es el usuario actual no fue ni coordinador ni responsable al entrar en el caso de uso. 4. El coordinador del proyecto debe ser PDI.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso). 2. Error responsable (si el usuario actual es Empleado Dpto y no es el responsable o el coordinador del proyecto que intenta modificar, se notificará un error y se proporcionará el nombre del actual responsable; termina el caso de uso). 3. Error aceptar (datos inválidos; volver al paso 4)
Casos de uso subordinados	**Designar Líneas** (en su caso) **Asignar Empleado** (en su caso) donde los empleados son: • el coordinador del proyecto • el responsable del proyecto Nota: **Notificar Interesados** (coordinador antiguo y nuevo, responsable antiguo y nuevo). **Designar Empleados** (en su caso) donde los empleados son: • los empleados que trabajan en el proyecto Nota: implica **Notificar Interesados** (trabajadores añadidos y eliminados) **Notificar Interesados** donde los interesados son: • el responsable del proyecto • el administrador

Caso de Uso	BORRAR PROYECTO
Descripción	El usuario elimina un proyecto del sistema. No hace falta borrar a los proyectos terminados; no se mostrarán entre los proyectos actuales de las páginas Web generadas si la fecha_final del contrato ya ha pasado. Además, se puede generar una página de antiguos proyectos.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El proyecto que se quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un proyecto de entre los proyectos existentes. 2. El usuario selecciona un proyecto. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma, el sistema elimina a) el proyecto. b) las relaciones que tiene el proyecto con otras entidades (empleados, eventos, líneas de investigación) 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Una línea de investigación no puede participar en un proyecto inexistente. 2. Un empleado no puede coordinar, trabajar oficialmente en, o ser responsable de un proyecto inexistente. 3. Un evento no puede ser anunciado por un proyecto inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el responsable del proyecto • el coordinador del proyecto • el administrador

10.2.1.4 Gestión de grupos de investigación

En el modelo de datos actual un grupo no tiene ni responsable (de los datos publicados en Web) ni coordinador.

Un grupo existente siempre se mostrará en las páginas Web generadas; quizás se debería añadir al modelo de datos un campo que indica si se quiere mostrar el grupo en las páginas Web o no. De este modo no haría falta borrar de la base de datos grupos que ya no existen.

Caso de Uso	CREAR GRUPO INV.
Descripción	El usuario da de alta un nuevo grupo de investigación.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo grupo de investigación. 2. El usuario introduce los datos, en particular: a) El usuario puede crear unas líneas de investigación del nuevo grupo (**Crear Línea Inv**). Además de las líneas creadas explícitamente por el usuario, se creará la línea de investigación por defecto del grupo. b) El titulo_abreviado que elige el usuario no puede coincidir con el de ningún grupo existente ni el de ninguna línea existente. *c) El usuario elige a un coordinador* (*que también se considera miembro*) *de entre los PDI, y posiblemente a un responsable* (*de los datos publicados en Web*) (***Asignar Empleado***). *Si no asigna ningún responsable, el sistema establecerá el propio usuario como responsable.* También puede elegir a más miembros (**Designar Empleados**), debe elegir a al menos uno si el coordinador no está implementado en el modelo de datos. 3. El usuario cancela o acepta la operación a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El grupo de investigación debe tener: un título, un título_abreviado, una descripción, un email, *un coordinador y un responsable* (*de los datos publicados en Web*).

	2. *El grupo de investigación debe tener al menos un miembro PDI (satisfecho si tiene coordinador*). 3. El grupo de investigación debe tener al menos una línea de investigación (satisfecho por la línea por defecto). 4. El coordinador del grupo debe ser PDI. 5. El título, el title (si existe), el título_abreviado el abbreviated_title (si existe) y la URL_logo del grupo de investigación deben ser únicos. 6. El título_abreviado del grupo de investigación no puede ser el mismo que el de cualquier línea de investigación (explicación: el título abreviado de la línea por defecto coincide con el de su grupo).
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1).
Casos de uso subordinados	**Crear Línea Inv** (en su caso) ***Asignar Empleado*** *donde los empleados son:* • *el responsable del grupo(por defecto, el usuario)* • *el coordinador del grupo* Nota: implica **Notificar Interesados** (*responsable y coordinador*). **Designar Empleados** (en su caso) donde los empleados son: • los miembros del proyecto Nota: implica **Notificar Interesados** (miembros) **Notificar Interesados** donde los interesados son: • el administrador.

Caso de Uso	MODIFICAR GRUPO INV.
Descripción	El usuario modifica los datos de un grupo de investigación.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo *Empleado Dpto* (*coordinador o responsable del grupo*)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El grupo que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un proyecto de entre los grupos existentes. 2. El usuario selecciona un grupo *3. Si el usuario tiene el perfil de Empleado Dpto* *a) Si no es ni coordinador ni responsable del grupo seleccionado, excepción* (*no se restringe con una precondición porque se quiere utilizar la excepción para comunicar el nombre del actual responsable, puesto que esta información no se publicará en las páginas Web*). 4. El sistema le ofrece la posibilidad de modificar los datos del grupo seleccionado. 5. El usuario introduce las modificaciones; en particular: a) El usuario puede modificar los datos de las líneas que componen el grupo (**Crear Línea Inv**, **Modificar Línea Inv** y **Borrar Línea Inv**), incluyendo la posibilidad de asignar una línea dada a otro grupo. b) El usuario puede modificar *el coordinador y el responsable* (***Asignar Empleado***) *y* los miembros (**Designar Empleados**). 6. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El grupo de investigación debe tener: un título, un título_abreviado, una descripción, un email, *un coordinador y un responsable.* También debe tener al menos un miembro PDI (*satisfecho si tiene coordinador*) y al menos una línea de investigación (satisfecho por la línea por defecto). 2. El título, el title (si existe), el título_abreviado el

	abbreviated_title (si existe) y la URL_logo del grupo de investigación deben ser únicos. 3. El título_abreviado del grupo de investigación no puede ser el mismo que el de cualquier línea de investigación. 4. El coordinador del grupo debe ser PDI. 5. Caso de un usuario con el perfil de Empleado Dpto: a) No puede haberse cambiado en la base de datos ningún dato de un grupo del que el empleado que es el usuario actual no fue ni coordinador ni responsable al entrar en el caso de uso.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) *2. Error responsable* (*si el usuario actual es Empleado Dpto y no es el responsable o el coordinador del grupo que intenta modificar, se notificará un error y se proporcionará el nombre del actual responsable; termina el caso de uso*). 3. Error aceptar (datos inválidos: volver al paso 4).
Casos de uso subordinados	**Crear Línea Inv** (en su caso) **Modificar Línea Inv** (en su caso) **Borrar Línea Inv** (en su caso) ***Asignar Empleado*** (*en su caso*) *donde los empleados son:* • *el responsable del grupo* • *el coordinador del grupo* *Nota: implica* ***Notificar Interesados*** (*coordinador antiguo y nuevo, responsable antiguo y nuevo*). **Designar Empleados** (en su caso) donde los empleados son: • los miembros añadidos y eliminados del proyecto Nota: implica **Notificar Interesados** (miembros añadidos y eliminados) **Notificar Interesados** donde los interesados son: • *el responsable del grupo.* • el administrador.

Caso de Uso	BORRAR GRUPO INV.
Descripción	El usuario elimina un grupo del sistema
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El grupo debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un grupo de entre los grupos existentes. 2. El usuario selecciona un grupo. 3. El usuario cancela o acepta la operación 4. El sistema pide confirmación (si no se ha cancelado), recordando que se eliminará también las líneas de investigación que componen el grupo. 5. Si el usuario confirma: a) el sistema le comunica que tiene que re-asignar a otro grupo los empleados que son miembros de éste antes de poder borrarlo y proporciona facilidades para hacerlo (*en el modelo de datos: un PDI tiene que ser miembro de un grupo y un grupo tiene que tener al menos un PDI como miembro*). 6. El sistema elimina a) el grupo, b) el resto de las relaciones que el grupo tiene con otras entidades (empleados, eventos), c) las lineas de investigación que componen el grupo d) las relaciones que estas líneas tienen con otras entidades (proyectos, en caso que el proyecto no esté asociado a otra línea, se deberá asignar el grupo a una nueva línea). 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Una línea de investigación no puede pertenecer a un grupo de investigación inexistente. 2. Un empleado no puede ser *ni coordinador, ni responsable, ni* miembro de un grupo de investigación inexistente. 3. Un evento no puede ser anunciado por un grupo de investigación inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso)

Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • *el responsable del grupo* • *el coordinador del grupo* • el administrador

Caso de Uso	CREAR LÍNEA INV.
Descripción	El usuario da de alta una nueva línea de investigación.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo *Empleado Dpto* (*coordinador o responsable del grupo*)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. *3. Si el perfil del usuario es Empleado Dpto, no puede crear una línea de investigación que pertenece a un grupo del que no es ni coordinador ni responsable.*
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos de la nueva línea. 2. El usuario introduce los datos, en particular: a) *El usuario puede elegir a los proyectos de investigación en los que la línea participa* (**Designar Proyectos**). b) El titulo_abreviado que elige el usuario no puede coincidir con el de ningún grupo existente ni el de ninguna línea existente. c) La creación de una línea sólo puede hacerse en el contexto de la creación o modificación de un grupo; por tanto, no hace falta elegir explícitamente su grupo. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La línea de investigación debe tener: un título, un título_abreviado, una descripción y un grupo. 2. El título, el title (si existe), el título_abreviado, el abbreviated_title (si existe), la URL_home_es (si existe) y la URL_home_en (si existe) de la línea de investigación deben ser únicos. 3. El título_abreviado de la línea de investigación no puede ser el mismo que el de cualquier grupo de investigación (explicación: el título abreviado de la línea por defecto coincide con el de su grupo). *4. Caso de un usuario con el perfil de Empleado Dpto:* a) *No puede haberse creado en la base de datos una línea que pertenece a un grupo del que el empleado que es el usuario actual no fue ni coordinador ni responsable al entrar en el caso de uso que hizo la llamada.*

Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	**Designar Proyectos** (en su caso). Nota: implica **Notificar Interesados** (responsable de cada proyecto). **Notificar Interesados** donde los interesados son: • *el responsable del grupo al que pertenece la línea* • el administrador

Caso de Uso	MODIFICAR LÍNEA INV.
Descripción	El usuario modifica los datos de una línea de investigación existente
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo *Empleado Dpto* (*coordinador o responsable del grupo*)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La línea de investigación que se quiere modificar debe estar dada de alta en el sistema. 4. *Si el perfil del usuario es Empleado Dpto, solo tiene acceso a los grupos de investigación de los que es coordinador o responsable y a las líneas de investigación que los componen.*
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una línea de entre las líneas a las que tiene acceso. 2. El usuario selecciona una línea. 3. El sistema le ofrece la posibilidad de modificar los datos de línea seleccionada. 4. El usuario introduce las modificaciones; en particular: a) El usuario puede modificar la lista de proyectos de investigación en los que la línea participa (**Designar Proyectos**). b) El usuario puede modificar el grupo al que pertenece la línea a otro al que tiene acceso (**Asignar Grupo**) 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La línea de investigación debe tener: un título, un título_abreviado, una descripción y un grupo. 2. El título, el title (si existe), el título_abreviado, el abbreviated_title (si existe), la URL_home_es (si existe) y la URL_home_en (si existe) de la línea de investigación deben ser únicos. 3. El título_abreviado de la línea de investigación no puede ser el mismo que el de cualquier grupo de investigación. *4. Caso de un usuario con el perfil de Empleado Dpto:* *a) No puede haberse cambiado en la base de datos ningún dato de una línea que pertenece a un grupo del que el*

	empleado que es el usuario actual no fue ni coordinador ni responsable al entrar en el caso de uso que hizo la llamada.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)
Casos de uso subordinados	**Designar Proyectos** (en su caso) Nota: implica **Notificar Interesados** (responsable de cada proyecto añadido o eliminado). **Asignar Grupo** (en su caso) *Nota: implica **Notificar Interesados*** (*responsable del grupo*). **Notificar Interesados** donde los interesados son: • *el responsable del grupo al que pertenece la línea* • el administrador

Caso de Uso	BORRAR LÍNEA INV.
Descripción	El usuario elimina una línea de investigación del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo *Empleado Dpto* (*coordinador o responsable del grupo*)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La línea de investigación que se quiere eliminar debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione la línea de entre las líneas existentes. 2. El usuario selecciona una línea (no se puede borrar la línea por defecto de un grupo de investigación). 3. El usuario cancela o acepta la operación 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma: a) Si la línea es el único participante en algún proyecto, excepción. 6. El sistema elimina b) la línea de investigación c) las relaciones de la línea con otras entidades (proyectos) 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Un grupo de investigación no puede tener una línea de investigación inexistente. 2. Un proyecto de investigación no puede tener como participante una línea de investigación inexistente. 3. Todo proyecto del sistema tiene que tener al menos una línea de investigación que participa en el. *4. Caso de un usuario con el perfil de Empleado Dpto:* a) *No puede haberse borrado de la base de datos una línea que pertenece a un grupo del que el empleado que es el usuario actual no fuera ni coordinador ni responsable al entrar en el caso de uso que hizo la llamada.*

Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado (el sistema comunica al usuario que tiene que escoger entre: a) relacionar el proyecto direcamente al grupo al que pertenece la línea, es decir, relacionarlo con la línea por defecto de este grupo b) borrar también el proyecto; volver al paso 6)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • *el responsable del grupo al que pertenece la línea* • el administrador

10.2.1.5 Gestión de publicaciones

Caso de Uso	CREAR PUBLICACIÓN
Descripción	El usuario da de alta una o varias publicaciones. Actualmente, solo se permite crear publicaciones a partir de un fichero con datos de publicaciones en formato BibTeXML. Se estudia la posibilidad de importar datos de / exportar datos a Universitas XXI y a DSpace para no tener que duplicar las operaciones con publicaciones en varias aplicaciones.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. Cada publicación que se quiere crear tiene que tener al menos un autor que es empleado del departamento.
Flujo Básico de Datos	1. El sistema pide al usuario que suba el fichero que contiene los datos de las publicaciones que se quiere crear. 2. El usuario sube el fichero. 3. Si el XML no es válido, excepción. 4. El sistema pide al usuario los datos siguientes: a) El usuario puede asignar un responsable (de los datos publicados en Web) por publicación del fichero proporcionado (**Asignar Empleado**); si no asigna ningún responsable, el sistema establecerá el propio usuario como responsable. b) El usuario puede asignar, por cada publicación, la URL de un fichero que contiene la publicación (ej. en formato pdf o postscript). c) *El usuario debe asignar los autores_locales de cada publicación del fichero proporcionado. Es decir, que debe indicar, por cada autor de cada publicación, la coincidencia, si hay, entre ese autor y un empleado del departamento* (**Asignar Empleado** o indicar 'autor no local'). Respecto al resto de los "datos de busqueda"[5] de la publicación, el sistema los extrae automáticamente del XML. Si en el XML faltan alguno de los "datos de búsqueda" obligatorios, excepción. 5. El usuario cancela o acepta la operación: a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor, p.e. una publicación sin ningún

[5] Los "datos de busqueda" de una publicación se definen actualmente como los atributos siguientes: title, keywords, año y autores_locales.

	autor_local), excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. La publicación debe tener: un título, un año, un responsable (de los datos publicados en Web), una entidadXML que contiene BibTeXML válido, y al menos un autor_local. 2. Los "datos de busqueda" de la publicación tienen que estar coherentes con los datos correspondientes del XML (valor del atributo entidadXML).
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error XML inválido (si el fichero no contiene BibTeXML válido, se notificará el error y se proporcionará la localización de conversores de otros formatos a BibTeXML; termina el caso de uso). *3. Error XML incompleto (si en el XML falta algún "dato de búsqueda" obligatorio, se lo indicará al usuario; termina le caso de uso).* 4. Error aceptar (datos inválidos; volver al paso 1).
Casos de uso subordinados	**Asignar Empleado** donde los empleados son: • los autores_locales de cada publicación creada • el responsable de cada publicación creada (por defecto, el usuario) Nota: implica **Notificar Interesados** (autores_locales y responsable de cada publicación). **Notificar Interesados** donde los interesados son: • el administrador

Caso de Uso	MODIFICAR PUBLICACIÓN
Descripción	El usuario modifica los datos de una publicación existente.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (responsable) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La publicación que se quiere modificar debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una publicación de entre las publicaciones existentes. 2. El usuario selecciona una publicación. a) Si no es responsable de la publicación seleccionada, excepción (no se restringe con una precondición porque se quiere utilizar la excepción para comunicar el nombre del actual reponsable, puesto que esta información no se publicará en las páginas Web). 3. El sistema ofrece al usuario la posibilidad de modificar los datos de la publicación: a) El usuario puede modificar el responsable (**Asignar Empleado**) y la URL de la publicación. *b) El usuario puede modificar el valor del atributo entidadXML; el sistema tendrá que re-validar el XML (si no es válido, excepción; se estudiará la posibilidad de integrar un editor de XML para evitar la excepción), extraer de nuevo los valores de los "datos de búsqueda" (si faltan datos obligatorios, excepción) y pedir al usuario que re-asigna los autores_locales* (**Asignar Empleado***); se estudiará la posibilidad de solo pedir al usuario que re-asigne los autores que han sido añadidos o modificados.* *c) Se estudiará la posibilidad de permitir al usuario modificar directamente el valor de los "datos de búsqueda". Pero en este caso, el sistema tendrá que modificar también los datos correspondientes del XML (valor del atributo entidadXML) para mantener la coherencia.* 4. El usuario introduce las modificaciones permitidas. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una*

	anotación al registro histórico. b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. La publicación debe tener: un título, un año, un responsable (de los datos publicados en Web), una entidadXML que contiene BibTeXML válido, y al menos un autor_local. 2. *Los "datos de busqueda" de la publicación tienen que estar coherentes con los datos correspondientes del XML (valor del atributo entidadXML).* 3. Caso de un usuario con el perfil de Empleado Dpto: a) No puede haberse cambiado en la base de datos ningún dato de una publicación de la que el empleado que es el usuario_actual no fue responsable al entrar en el caso de uso.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error XML inválido (si el BibTeXML modificado no es válido, se notificará el error; volver al paso 3). 3. Error XML incompleto (si en el XML falta algún "dato de búsqueda" obligatorio, se lo indicará al usuario; termina el caso de uso). 4. Error aceptar (datos inválidos; volver al paso 3). 5. Error responsable (si el usuario actual es Empleado Dpto y no es el responsable de la publicación que intenta modificar, se notificará un error y se proporcionará el nombre del actual responsable; termina el caso de uso).
Casos de uso subordinados	**Asignar Empleado** donde los empleados son: • los autores_locales • el responsable Nota: implica **Notificar Interesados** (autores_locales nuevos y antiguos y responsable nuevo y antiguo). **Notificar Interesados** donde los interesados son: • el responsable de la publicación • el administrador

Caso de Uso	BORRAR PUBLICACIÓN
Descripción	El usuario quiere eliminar una publicación del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (responsable) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La publicación que se quiere borrar debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una publicación de entre las publicaciones existentes. 2. El usuario selecciona una publicación. a) Si no es responsable de la publicación seleccionada, excepción (no se restringe con una precondición porque se quiere utilizar la excepción para comunicar el nombre del actual reponsable, puesto que esta información no se publicará en las páginas Web). 3. El usuario cancela o acepta la operación 4. El sistema pide confirmación (si no se ha cancelado). 5. Si el usuario confirma, el sistema elimina: a) la publicación b) las relaciones de la publicación con otros entidades (empleados) 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. Un empleado no puede ser ni autor_local ni responsable de una publicación inexistente. 2. Caso de un usuario con el perfil de Empleado Dpto: a) No puede haberse cambiado en la base de datos ningún dato de la que el empleado que es el usuario_actual no fue responsable al entrar en el caso de uso.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error responsable (si el usuario actual es Empleado Dpto y no es el responsable de la publicación que intenta borrar, se notificará un error y se proporcionará el nombre del actual responsable; termina el caso de uso).
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el responsable • el administrador

10.2.1.6 Gestión de tesis

La información sobre las tesis del departamento no se publicará en las páginas Web del departamento sino en las páginas Web de los grupos de investigación del departamento. En el modelo de datos actual, sólo se contemplan dos tipos de tesis: PFC y doctorados (no se contemplan tesis de master). Las tesis se relacionan con los grupos a través del grupo del director (de interés para la generación de las páginas Web de los grupos).

Caso de Uso	CREAR TESIS
Descripción	El usuario da de alta una tesis. Nota: un PFC no tiene autor_local; una tesis doctoral puede, o no, tener autor_local.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (director de la tesis) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos. 2. El usuario introduce los datos; en particular: a) Primero, el usuario elige el tipo de tesis. *b1) Si el perfil del usuario es Empleado Dpto. el sistema asigna al propio usuario como director.* *b2) Si el perfil del usuario no es Empleado Dpto, el sistema le pide que eliga a un director de entre los PDI doctores, si se trata de una tesis doctoral, o de entre los PDI, si se trata de un PFC (**Asignar Empleado**).* c) Si se trata de una tesis doctoral, el usuario puede eligir a un autor_local de entre los empleados (**Asignar Empleado**). 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La tesis debe tener un título, un autor y un director (pero no necesariamente un autor_local). 2. El URL_pdf (si existe) y el autor_local (si existe) de la tesis deben ser únicos. 3. Si se trata de una tesis doctoral, el director debe ser doctor y si se trata de un PFC, el director debe ser PDI. 4. Si el autor_local (es decir, autor de entre los empleados dados de alta en el sistema) existe, el valor del atributo

	autor es el nombre de este empleado. 5. Caso de un usuario que tiene el perfil de Empleado Dpto: *a) No puede haberse creado en la base de datos una nueva tesis que no tenga como director el empleado que es el usuario actual.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1).
Casos de uso subordinados	**Asignar Empleado** donde los empleados son: • el director, • el autor_local (en su caso) Nota: implica **Notificar Interesados** (director, autor_local) **Notificar Interesados** donde los interesados son: • el administrador

Caso de Uso	MODIFICAR TESIS
Descripción	El usuario modifica los datos de una tesis.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (director o autor_local de la tesis) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La tesis que se quiere modificar debe estar dada de alta en el sistema. 4. Si el perfil de usuario es Empleado Dpto solo tiene acceso a las tesis de las que es o bien director o bien autor_local.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una tesis de entre las tesis a las que tiene acceso. 2. El usuario selecciona una tesis. *3A. Si el perfil del usuario no es Empleado Dpto o es Empleado Dpto y se trata del director de la tesis* *a) El sistema le ofrece la posibilidad de modificar todos los datos de la tesis.* *3B. Si el perfil del usuario es Empleado Dpto y se trata del autor_local de una tesis doctoral* *a) El sistema le ofrece la posibilidad de modificar todos los datos de la tesis menos los "datos restringidos"[6].* 4. El usuario introduce las modificaciones, en particular: a) El usuario puede modificar el director (**Asignar Empleado**) y, en el caso de una tesis doctoral, el autor_local (**Asignar Empleado**), si tiene el perfil adecuado. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. La tesis debe tener un título, un autor y un director (pero no necesariamente un autor_local). 2. El URL_pdf (si existe) y el autor_local (si existe) de la tesis deben ser únicos. 3. Si se trata de una tesis doctoral, el director debe ser doctor y

[6] Los "datos restringidos" de una tesis se definen actualmente como los atributos siguientes: director y autor_local.

	si se trata de un PFC, el director debe ser PDI. 4. Si el autor_local (es decir, autor de entre los empleados dados de alta en el sistema) existe, el valor del atributo autor es el nombre de este empleado. 5. *Caso de un usuario que tiene el perfil de Empleado Dpto:* *a) No puede haberse cambiado en la base de datos los datos de una tesis del que el empleado que es el usuario actual no sea ni director ni autor_local.* b) *Si la tesis es una tesis doctoral y el usuario es el autor_local, no puede haberse cambiado en la base de datos ni el director y ni el autor_local.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1).
Casos de uso subordinados	**Asignar Empleados** (en su caso) donde los empleados son: • el director nuevo • el autor_local nuevo Nota: implica **Notificar Interesados** (director nuevo y antiguo, autor_local nuevo y antiguo) **Notificar Interesados** donde los interesados son: • el director • el administrador

Caso de Uso	BORRAR TESIS
Descripción	El usuario elimina una tesis del sistema. No se debe borrar a las tesis terminadas; no se mostrarán entre las tesis en curso de las páginas Web generadas si tiene fecha de lectura y ya ha pasado esa fecha.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (director de la tesis) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La tesis que se quiere borrar debe estar dada de alta en el sistema. 4. Si el perfil de usuario es Empleado Dpto solo tiene acceso a las tesis de las que es director.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione una tesis de entre las tesis a las que tiene acceso. 2. El usuario selecciona una tesis. 3. El usuario cancela o acepta la operación 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma, el sistema elimina a) la tesis b) las relaciones que tiene la tesis con otras entidades del sistema (empleados). 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. Un empleado no puede ser ni director ni autor_local de una tesis inexistente. *2. Caso de un usuario que tiene el perfil de Empleado Dpto:* a) *No puede haberse borrado de la base de datos una tesis del que el empleado que es el usuario actual no fuera el director*.
Excepciones	1. Error Interno (ej. conexión a BBDD; termina el caso de uso)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el director • el administrador

10.2.1.7 Gestión de eventos

Caso de Uso	CREAR EVENTO
Descripción	El usuario da de alta un nuevo evento (noticia, enlace, etc...).
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo evento. 2. El usuario introduce los datos, en particular: a) El usuario elige una o varias categorías para el evento de entre las categorías existentes (que definen en qué páginas se publicará el evento). *b) El usuario puede asignar el evento a uno o varios proyectos de investigación (**Designar Proyectos**) o grupos de investigación (**Designar Grupos**).* 3. El usuario cancela o acepta la operación a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. El evento debe tener: un título, un título abreviado, una URL_es, una fecha, *y una o varias categorías*. 2. El titulo_abreviado del evento debe ser único.
Excepciones	1. Error Interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	***Designar Proyectos*** (*en su caso*). *Nota: implica **Notificar Interesados*** (*responsable de cada proyecto*). ***Designar Grupos*** (*en su caso*). *Nota: implica **Notificar Interesados*** (*responsable de cada grupo*). **Notificar Interesados** donde los interesados son: 1. el administrador

Caso de Uso	MODIFICAR EVENTO
Descripción	El usuario modifica los datos de un evento existente.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El evento que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un evento de entre los eventos existentes. 2. El usuario selecciona un evento. 3. El sistema ofrece al usuario la posibilidad de modificar los datos del evento seleccionado. 4. El usuario introduce las modificaciones, en particular: a) El usuario puede modificar la lista de proyectos de investigación (**Designar Proyectos**) o grupos de investigación (**Designar Grupos**) que anuncian el evento. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.* b) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	1. El evento debe tener: un título, un título abreviado una URL_es, una fecha, y una o varias categorías. 2. El titulo_abreviado del evento debe ser único.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)
Casos de uso subordinados	**Designar Proyectos** (en su caso). Designar Proyectos (en su caso). *Nota: implica* ***Notificar Interesados*** (*responsable de cada proyecto añadido o eliminado*). **Designar Grupos** (en su caso). *Nota: implica* ***Notificar Interesados*** (*responsable de cada grupo añadido o eliminado*). **Notificar Interesados** donde los interesados son: • el administrador

Caso de Uso	BORRAR EVENTO
Descripción	El usuario elimina un evento del sistema. No hace falta borrar a los eventos pasados; no se mostrarán en las páginas Web generadas si su fecha ha pasado.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El evento debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un evento de entre los eventos existentes. 2. El usuario selecciona un evento. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma, el sistema elimina a) el evento. b) las relaciones del evento con otras entidades (proyectos y grupos) 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*. b) *Si avisos está activado*, **Notificar Interesados**.
Postcondiciones	1. Un grupo de investigación no puede anunciar un evento inexistente. 2. Un proyecto de investigación no puede anunciar un evento inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el administrador.

10.2.1.8 Gestión de despachos

Caso de Uso	CREAR DESPACHO
Descripción	El usuario crea un nuevo despacho de trabajo.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo despacho. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: b) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. El despacho debe tener un nombre, una extensión y un campus. 2. El nombre y la extensión del despacho deben ser únicos.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1).

Caso de Uso	MODIFICAR DESPACHO
Descripción	El usuario quiere modificar los datos de un despacho del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El despacho que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un despacho de entre los despachos existentes. 2. El usuario elige un despacho. 3. El sistema ofrece al usuario la posibilidad de modificar los datos del despacho. 4. El usuario introduce los datos, en particular: a) El usuario puede modificar los empleados que ocupan el despacho (**Designar Empleados**). b) El usuario no puede modificar el campus del despacho. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: b) *Si logging está activado, el sistema añade una anotación al registro histórico*. c) *Si avisos está activado,* **Notificar Interesados**.
Postcondiciones	3. El despacho debe tener un nombre, una extensión y un campus. 4. El nombre y la extensión del despacho deben ser únicos.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)
Casos de uso subordinados	**Designar Empleado** (en su caso) donde los empleados son: • los nuevos ocupantes del despacho. Nota: implica **Notificar Interesados** (nuevos y antiguos ocupantes) **Notificar Interesados** donde los interesados son: • el administrador

Caso de Uso	BORRAR DESPACHO
Descripción	El usuario elimina un despacho del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El despacho que se quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un despacho de entre los despachos existentes. 2. El usuario elige un despacho. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado). 5. Si el usuario confirma: a) Si el despacho tiene ocupantes, excepción 6. El usuario elimina: a) el despacho, b) las relaciones que tiene este despacho con otras entidades (campus). 7. Si la operación termina con éxito: b) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. Un empleado no puede estar ocupando un despacho inexistente.
Excepciones	1. Error Interno (Conexión a BBDD.Ej.) 2. Error borrado (el sistema comunica al usuario que tiene que reasignar un despacho a todos los empleados que ocupan actualmente al despacho antes de poder borrarlo; volver al paso 6)

Caso de Uso	CREAR CAMPUS
Descripción	El usuario crea un nuevo campus.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo campus. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. El campus debe tener un nombre y un prefijo. 2. El nombre y el prefijo del campus deben ser únicos
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

Caso de Uso	MODIFICAR CAMPUS
Descripción	El usuario quiere modificar los datos de un campus existente.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El campus que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un campus de entre los campus existentes. 2. El usuario elige un campus. 3. El sistema ofrece al usuario la posibilidad de modificar los datos del campus. 4. El usuario introduce los datos. 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. El campus debe tener un nombre y un prefijo. 2. El nombre y el prefijo del campus deben ser únicos
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)

Caso de Uso	BORRAR CAMPUS
Descripción	El usuario elimina un campus del sistema.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El campus que se quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un campus de entre los campus existentes. 2. El usuario elige un campus. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado). 5. Si el usuario confirma a) Si el campus tiene despachos, excepción 6. El usuario elimina el campus. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. Un despacho no puede estar localizado en un campus inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado (el sistema comunica al usuario que tiene que borrar todos los despachos de este campus antes de poder borrarlo y da facilidades para hacerlo; volver al paso 6).

10.2.1.9 Gestión de Home Dpto.

Se podría quitar el caso de uso Crear Home Dpto y crear la información necesaria directamente en la base de datos a la hora de desplegar el sistema pero este podría ser un caso especial ya que durante el tiempo de vida de la aplicación podrían aparecer cambios en los datos de la home que el administrador podría modificar directamente en la aplicación sin tener que atacar una b.b.d.d. directamente.

Caso de Uso	CREAR HOME DPTO
Descripción	El usuario crea los datos básicos del departamento. Esta operación se hace una sóla vez como parte del bootstrap del sistema.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La home_dpto no debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos básicos del departamento. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. La home_dpto debe tener un nombre, una dirección, un teléfono y un e-mail.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

Caso de Uso	MODIFICAR HOME DPTO
Descripción	El usuario quiere modificar los dátos del departamento.
Prioridad	1 (Primera fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La home_dpto debe estar dada de alta en el sistema.
Flujo Básico de Datos	1. El sistema ofrece al usuario la posibilidad de modificar los datos de la home_dpto. 2. El usuario introduce los datos, en particular: a) El usuario puede introducir XHTML directamente para el valor de los tres atributos siguientes: accesos, mapas y alojamiento. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos o sin haber modificado ningún campo, excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. La home_dpto debe tener un nombre, una dirección, un teléfono y un e-mail. *2. Cuando se añade la cabecera apropiada, el XHTML que constituye el valor de los atributos accesos, mapas y alojamiento tiene que validar contra la especificacion XHTML 1.1.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) *2. Error aceptar (datos inválidos; caso particular: el valor de los atributos no es válido con respecto a la especificación XHTML 1.1; volver al paso 1)*

10.2.1.10 Relaciones Comunes

Caso de Uso	ASIGNAR EMPLEADO
Descripción	El usuario elige a un empleado existente con el fin de crear una determinada relación entre éste y una determinada entidad. Por tanto, la entidad y la relación son parámetros del caso de uso. Actualmente, las relaciones–entidades posibles son: '–es coodinador de–una asignatura', '–es responsable de–un proyecto', '–coordina–un proyecto', '–es responsable de–una publicación', '–es director de–una tesis', '–es autor_local de–una tesis' y '–ocupa–un cargo'. *Otras posibilidades que actualmente no existen en el modelo de datos pero que convendrían añadir son:* '–es *responsable de–un grupo'*, '–es *coordinador de–un grupo'*.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo Empleado Dpto (condiciones heredadas del caso de uso llamante)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El empleado que se quiere asociar a la entidad debe estar dado de alta en el sistema. 4. Si el usuario tiene el perfil de Empleado Dpto debe cumplir las condiciones impuestas por el caso de uso llamante.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un empleado de entre los empleados existentes que no están ya relacionados con la entidad y que satisfacen una cierta restricción (ejemplo de posible restricción: el coordinador de una asignatura debe ser doctor) 2. El usuario selecciona un empleado 3. El usuario cancela o acepta la operación. a) Si se acepta sin haber seleccionado un empleado, excepción. b) Crear la relación entre la entidad y un empleado implica eliminar una relación previa del mismo tipo que tiene esta entidad con otro empleado, en caso de existir. 4. Si la operación termina con éxito: a) *Si avisos está activado,* **Notificar Interesados**. b) Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.
Postcondiciones	1. Caso de un usuario que tiene el perfil de Empleado Dpto a) No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestion entre una entidad del tipo en cuestión y un empleado si implica incumplir

	las condiciones heredadas del caso de uso que hizo la llamada.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el empleado asociado y (en su caso) el empleado desasociado.

Caso de Uso	DESIGNAR EMPLEADOS
Descripción	El usuario elige uno o varios empleados existentes con el fin de crear o destruir una determinada relación entre éstos y una determinada entidad. Por tanto, la entidad y la relación son parámetros del caso de uso. Actualmente, las relaciones–entidades posibles son: '–son autores de–una publicación', '–trabajan en–un proyecto', '–imparten–una asignatura', '–son miembros de–un grupo', '–ocupan–un despacho'.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo Empleado Dpto (condiciones heredadas del caso de uso llamante)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. Los empleados que se quieren asociar a la entidad deben estar dados de alta en el sistema. 4. Si el usuario tiene el perfil de Empleado Dpto debe cumplir las condiciones impuestas por el caso de uso que hizo la llamada.
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione uno o varios empleados de entre los empleados existentes que no están ya relacionados con la entidad y que satisfacen una cierta restricción, con el fin de relacionarlos (ejemplo de posible restricción: el profesor de una asignatura debe ser PDI) b) y/o que seleccione uno o varios empleados de entre los empleados que están ya relacionados con la entidad con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina uno o varios empleados de la lista de empleados relacionados con la entidad. 3. El usuario cancela o acepta la operación. a) Si se acepta sin haber ni añadido ni eliminado ningún empleado, excepción. 4. Si la operación termina con éxito: a) *Si avisos está activado,* **Notificar Interesados**. *b) Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.*
Postcondiciones	2. Caso de un usuario que tiene el perfil de Empleado Dpto *a) No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestion entre una entidad del tipo en cuestión y un empleado si implica incumplir las condiciones heredadas del caso de uso llamante.*

Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • los empleados añadidos (en su caso) y eliminados (en su caso)

Caso de Uso	ASIGNAR GRUPO
Descripción	El usuario elige un grupo existente con el fin de crear una determinada relación entre éste y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente, las entidades-relaciones posibles son: 'una línea de investigación–pertenece a–' y 'un empleado–es miembro de–'
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo Empleado Dpto (responsable o coordinador del grupo)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El grupo al que se quiere asociar la entidad debe estar dado de alta en el sistema. 4. Si el perfil del usuario es Empleado Dpto. solo tiene acceso a los grupos de los que es responsable o coordinador.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un grupo de entre los grupos a los que tiene acceso que están ya relacionados con la entidad en cuestión. 2. El usuario selecciona un grupo. 3. El usuario cancela o acepta la operación. a) Si se acepta sin haber seleccionado un grupo, excepción. b) Crear la relación entre un grupo y la entidad implica eliminar una previa relación del mismo tipo que tiene esta entidad con otro grupo, en caso de existir. 4. Si la operación termina con éxito: *a) Si avisos está activado, Notificar Interesados.* b) *Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.*
Postcondiciones	1. Caso de un usuario que tiene el perfil de Empleado Dpto: *a) No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestión entre una entidad del tipo en cuestión y un grupo del que el empleado que es el usuario actual no es ni responsable ni coordinador.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	Notificar Interesados donde los interesados son: • el responsable del grupo seleccionado

Caso de Uso	DESIGNAR GRUPOS
Descripción	El usuario elige uno o varios grupos existentes con el fin de crear una determinada relación entre éstos y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente, la única relación–entidad posible es: '–anuncian–un evento'.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo *Empleado Dpto* (*responsable o coordinador del grupo*)[7]
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. Los grupos a los que se quiere asociar la entidad deben estar dados de alta. *4. Si el perfil del usuario es Empleado Dpto. solo tiene acceso a los grupos de los que es responsable o coordinador.*
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione uno o varios grupos de entre los grupos a los que tiene acceso que no están ya relacionados con la entidad, con el fin de relacionarlos b) y/o que seleccione uno o varios grupos de entre los grupos a los que tiene acceso que están ya relacionados con la entidad, con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina uno o varios grupos de la lista de grupos relacionados con la entidad. 3. El usuario cancela o acepta la operación. a) Si se acepta sin haber ni añadido ni eliminado ningún grupo, excepción. 4. Si la operación termina con éxito: *a) Si avisos está activado*, **Notificar Interesados**. b) *Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.*
Postcondiciones	1. *Caso de un usuario que tiene el perfil de Empleado Dpto*: a) *No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestión entre una entidad del tipo en cuestión y un grupo del que el empleado que es el usuario actual no es ni responsable ni coordinador.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

[7] Ni la relación de responsable (de los datos publicados en Web) de grupo ni el de coordinador de grupo existen en el modelo de datos actual.

Casos de uso subordinados	***Notificar Interesados*** *donde los interesados son:* • *el responsable de cada grupo añadido o eliminado*

Caso de Uso	DESIGNAR LÍNEAS
Descripción	El usuario elige uno o varias líneas de investigación existentes con el fin de crear una determinada relación entre éstas y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente, la única entidad–relación posible es un proyecto–pertenece a.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo Empleado Dpto (restricciones heredadas del caso de uso llamante)
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La línea debe estar dada de alta en el sistema. 4. Si el usuario tiene el perfil de Empleado Dpto tiene acceso a las líneas según las restricciones impuestas por el caso de uso que hizo la llamada.
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione una o varias líneas de investigación de entre las líneas a las que tiene acceso que no están relacionadas con la entidad, con el fin de relacionarlas. b) y/o que seleccione una o varias líneas de investigación de entre las líneas a las que tiene acceso que están relacionadas con la entidad, con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina una o varias líneas de la lista de líneas relacionadas con la entidad. 3. El usuario acepta o cancela la operación a) Si se acepta haber ni añadido ni eliminado ninguna línea de investigación, excepción. *b) Si se asigna un proyecto a una línea de un grupo y el proyecto ya estaba asignado a la línea por defecto del mismo grupo, al aceptar, se elimina la asociación con la línea por defecto.* *4. Si la operación termina con éxito:* a) *Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta*
Postcondiciones	*1. Un proyecto asignado a la línea de investigación por defecto de un grupo no puede estar asignado a otra línea de investigación del mismo grupo.* 2. Caso de un usuario que tiene el perfil de Empleado Dpto *a) No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestion entre una entidad del tipo en cuestión y un proyecto si implica incumplir las condiciones heredadas del caso de uso que hizo la llamada.*

Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	Ninguno

Caso de Uso	DESIGNAR PROYECTOS
Descripción	El usuario elige uno o varios proyectos existentes con el fin de crear una determinada relación entre éstos y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente las relaciones–entidades posibles es: '–anuncian–un evento', y '–tienen como participante–una línea'.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto (responsable o coordinador del proyecto) Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El proyecto debe estar dado de alta en el sistema. 4. Si el perfil del usuario es Empleado Dpto. solo tiene acceso a los proyectos de los que es responsable o coordinador.
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione uno o varios proyectos de entre los proyectos a los que tiene acceso que no están ya relacionados con la entidad, con el fin de relacionarlos b) y/o que seleccione uno o varios proyectos de entre los proyectos a los que tiene acceso que están ya relacionados con la entidad, con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina uno o varios proyectos de la lista de proyectos relacionados con la entidad. 3. El usuario cancela o acepta la operación. *a) Si se acepta sin haber ni añadido ni eliminado ningún proyecto, excepción.* 4. Si la operación termina con éxito: a) *Si avisos está activado,* **Notificar Interesados**. *b) Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.*
Postcondiciones	1. Caso de un usuario que tiene el perfil de Empleado Dpto: *a) No puede haberse ni creado ni destruido en la base de datos una relación del tipo en cuestión entre una entidad del tipo en cuestión y un proyecto del que el empleado que es el usuario actual no es ni responsable ni coordinador.*
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)
Casos de uso subordinados	**Notificar Interesados** donde los interesados son: • el responsable de cada proyecto añadido o eliminado.

Caso de Uso	DESIGNAR TITULACIONES
Descripción	El usuario elige una o varias titulaciones existentes con el fin de crear una determinada relación entre éstas y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente la única relacion–entidad posible es: 'asignatura–pertenece a–'. En el modelo de datos, no existe un director de la titulación ya que esta figura podría ser un empleado de otro departamento. Por tanto, no se notifica a nadie.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La titulación debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione una o varias titulaciones de entre las titulaciones que no están ya relacionadas con la entidad, con el fin de relacionarlas, b) y/o que seleccione una o varias titulaciones de entre las titulaciones que están ya relacionadas con la entidad, con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina una o varias titulaciones de la lista de titulaciones relacionadas con la entidad. 3. El usuario cancela o comete la operación. b) Si se comete sin haber ni añadido ni eliminado ninguna titulación, excepción. 4. Si la operación termina con éxito: c) Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.
Postcondiciones	Ninguna
Excepciones	3. Error interno (ej. conexión a BBDD; termina el caso de uso) 4. Error cometer (datos inválidos; volver al paso 1)
Casos de uso subordinados	Ninguno

Caso de Uso	DESIGNAR DESPACHOS
Descripción	El usuario elige uno o varios despachos existentes con el fin de crear una determinada relación entre éstos y una determinada entidad. Por tanto, la relación y la entidad son parámetros del caso de uso. Actualmente la única relacion–entidad posible es: 'empleado–ocupa–'. No se notifica a nadie.
Prioridad	1 (Primera fase)
Actores	Administrador Personal Administrativo
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. La titulación debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario a) que seleccione uno o varios despachos de entre los despachos que no están ya relacionados con la entidad, con el fin de relacionarlos, b) y/o que seleccione uno o varios despachos de entre los despachos que están ya relacionados con la entidad, con el fin de eliminar las correspondientes relaciones. 2. El usuario añade y/o elimina uno o varios despachos de la lista de despachos relacionados con la entidad. 3. El usuario cancela o comete la operación. a) Si se comete sin haber ni añadido ni eliminado ningún despacho, excepción. 4. Si la operación termina con éxito: a) Nota: el caso de uso que llama al presente se ocupa de añadir una anotación al registro histórico, si hace falta.
Postcondiciones	Ninguna
Excepciones	5. Error interno (ej. conexión a BBDD; termina el caso de uso) 6. Error cometer (datos inválidos; volver al paso 1)
Casos de uso subordinados	Ninguno

10.2.1.11 Gestión de tipos enumerados

Por la naturaleza de estas entidades, la adición o supresión de un tipo de perfil, de un tipo de contrato o de un tipo de evento implicaría cambios significativos en la aplicación. Por tanto, no tiene mucho sentido poder modificarlos o borrarlos a través de la interfaz por lo que no se definen casos de uso para ello. La creación se haría directamente en la base de datos. Sin embargo, y sólo para esta primera iteración de la aplicación, se ha añadido esta posibilidad dentro de la aplicación ya que el coste de realización en Ruby On Rails era mínimo y se quería estudiar el comportamiento de la aplicación ante estos cambios. Además, se ha pensado que para el prototipo, sería conveniente tener dominio sobre todas las operaciones sobre b.b.d.d. de la aplicación. Por esto, aunque en este anexo no se incluyen los casos de uso citados, en el capítulo 4 de esta memoria si que se hace un pequeño estudio del funcionamiento de estos.

En lo que respecta a los cargos y a los tipos de proyecto, sin embargo, sí que tiene sentido permitir la modificación e incluso eliminacón por interfaz.

En el modelo de datos actual, una persona no puede tener varios cargos y no se distingue entre cargos unipersonales y multipersonales. Obsérvese que un empleado no tiene por qué tener un cargo y, en el modelo actual, un cargo no tiene por qué estar ocupado por un empleado.

Caso de Uso	CREAR TIPO CARGO
Descripción	El usuario da de alta un nuevo cargo.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo cargo. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. El cargo debe tener un nombre único.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

Caso de Uso	MODIFICAR TIPO CARGO
Descripción	El usuario modifica los datos de un cargo existente.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El cargo que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un cargo de entre los cargos existentes. 2. El usuario selecciona un cargo. 3. El sistema ofrece al usuario la posibilidad de modificar los datos del cargo. 4. El usuario introduce las modificaciones; en particular: *a) El usuario puede cambiar el empleado* (***Asignar Empleado***) *o empleados* (***Designar Empleados***) *– dependiendo de si el cargo es unipersonal o multipersonal – que tiene*(*n*) *asignado*(*s*) *el cargo actualmente* 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos *o sin haber modificado ningún campo*, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. El cargo debe tener un nombre único.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)
Casos de uso subordinados	***Asignar Empleado*** (*en su caso*) *donde el empleado es*: • *el nuevo ocupante del cargo.* *Nota: implica* ***Notificar Interesados*** (*nuevo y antiguo ocupante*) ***Designar Empleado*** (*en su caso*) *donde los empleados son*: • *los nuevos ocupantes del cargo.* *Nota: implica* ***Notificar Interesados*** (*nuevos y antiguos ocupantes*).

Caso de Uso	BORRAR TIPO CARGO
Descripción	El usuario elimina un cargo del sistema. Si un cargo no está ocupado, no significa que se tiene que borrar.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El cargo que es quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un cargo de entre los cargos existentes 2. El usuario selecciona un cargo. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma: a) si el cargo está ocupado, excepción 6. El sistema elimina el cargo. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. Un empleado no puede estar ocupando un cargo inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado (el sistema comunica al usuario que tiene que reasignar el cargo a otro(s) empleado(s) antes de poder eliminarlo y ofrece facilidades para hacerlo; volver al paso 6).

Caso de Uso	CREAR TIPO PROYECTO
Descripción	El usuario da de alta un nuevo tipo de proyecto.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado.
Flujo Básico de Datos	1. El sistema pide al usuario que introduzca los datos del nuevo tipo de proyecto. 2. El usuario introduce los datos. 3. El usuario cancela o acepta la operación. b) Si se acepta con datos inválidos (incluyendo campos obligatorios sin valor), excepción. 4. Si la operación termina con éxito: b) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. El tipo de proyecto debe tener un nombre único.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 1)

Caso de Uso	MODIFICAR TIPO PROYECTO
Descripción	El usuario modifica los datos de un tipo de proyecto existente.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El tipo de proyecto que se quiere modificar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un tipo de proyecto de entre los tipos existentes. 2. El usuario selecciona un tipo de proyecto. 3. El sistema ofrece al usuario la posibilidad de modificar los datos del tipo de proyecto. 4. El usuario introduce las modificaciones, en particular: a) *El usuario puede cambiar los proyectos que actualmente tienen este tipo (**Designar Proyectos**). Si añade un proyecto a la lista de proyectos que tienen este tipo, el proyecto pierde su tipo antiguo. Si elimina un proyecto de la lista de proyectos que tienen este tipo, tiene que asignarle otro tipo.* 5. El usuario cancela o acepta la operación. a) Si se acepta con datos inválidos *o sin haber modificado ningún campo*, excepción. 6. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico*.
Postcondiciones	1. El tipo de proyecto debe tener un nombre único. 2. Todo proyecto debe tener un tipo.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error aceptar (datos inválidos; volver al paso 3)
Casos de uso subordinados	**Designar Proyectos** (en su caso) Nota: implica Notificar Interesados (responsable de cada proyecto añadido o eliminado).

Caso de Uso	BORRAR TIPO PROYECTO
Descripción	El usuario elimina un tipo de proyecto del sistema. Si un tipo de proyecto no tiene asociado ningún proyecto, no significa que se tiene que borrar.
Prioridad	2 (Segunda fase)
Actores	Administrador
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El tipo de proyecto que es quiere borrar debe estar dado de alta en el sistema.
Flujo Básico de Datos	1. El sistema pide al usuario que seleccione un tipo de proyecto de entre los tipos existentes 2. El usuario selecciona un tipo de proyecto. 3. El usuario cancela o acepta la operación. 4. El sistema pide confirmación (si no se ha cancelado) 5. Si el usuario confirma: a) si existen proyectos con este tipo, excepción 6. El sistema elimina el tipo de proyecto. 7. Si la operación termina con éxito: a) *Si logging está activado, el sistema añade una anotación al registro histórico.*
Postcondiciones	1. Un proyecto no puede tener un tipo inexistente.
Excepciones	1. Error interno (ej. conexión a BBDD; termina el caso de uso) 2. Error borrado (el sistema comunica al usuario que tiene que reasignar los proyectos que tienen este tipo a otro tipo antes de poder eliminarlo y ofrece facilidades para hacerlo; volver al paso 6).

10.2.1.12 Utilidades

Caso de Uso	NOTIFICAR INTERESADOS
Descripción	El sistema envía correos electrónicos a los usuarios que intervienen en la opercación. El tipo de correo generado depende de la operación realizada que, por tanto, es un parámetro del caso de uso.
Prioridad	1 (Primera fase)
Actores	Administrador Empleado Dpto Personal Administrativo *Técnico*
Precondiciones	1. El usuario debe estar dado de alta en el sistema. 2. El usuario debe estar logado. 3. El sistema debe poseer los correos de los interesados.
Flujo Básico de Datos	1. El sistema comprueba los usuarios que intervienen en la operación *y descarta el usuario actual si se encuentra entre ellos*. 2. El sistema recupera sus correos 3. El sistema envía un correo a los interesados; el formato de este correo depende del tipo de usuario y de la operación realizada.
Postcondiciones	Ninguna.
Excepciones	1. Error Interno (ej. conexión a BBDD; termina el caso de uso) 2. Error de Correo (no se ha podido enviar el correo; termina el caso de uso)

Caso de Uso	OBTENER WEB PERSONAL
Descripción	Un Empleado Dpto accede al sistema para obtener fichero(html,xml,etc..) autogenerado al dar de alta al empleado
Prioridad	1 (Primera fase)
Actores	Empleado Dpto
Precondiciones	1. El empleado debe estar dado de alta en el sistema. 2. El empleado debe estar logado.
Flujo Básico de Datos	1. El empleado accede a su sección personal. 2. El empleado obtiene el fichero (XML en esta versión).
Postcondiciones	Ninguna.
Excepciones	1. Error Interno (Conexión a BBDD.Ej.) 2. Error de Documento.

10.3 Anéxo C.

10.3.1 Montando Ruby on Rails en Debian.

- Instalamos los paquetes para Ruby:

  ```
  $sudo apt-get install aptitude install ruby libzlib-ruby rdoc irb
  $wget http://rubyforge.org/frs/download.php/1399/rubygems-0.8.1.tgz
  $tar -zxvf rubygems-0.8.1.tgz
  $cd rubygems-0.8.1
  $sudo ruby1.8 install.rb
  ```

- Instalamos rails:

  ```
  $sudo gem install rails --include-dependencies
  ```

- Creamos el proyecto:

  ```
  $rails webdep
  ```

- Cambiamos al directorio creado:

  ```
  $cd webdep
  ```

- Para probar la configuración de Ruby iniciamos el RoR server (WEBrick):

  ```
  $ruby script/server
  ```

  Debemos ver algo como esto:

  ```
  => Booting WEBrick...
  => Rails application started on http://0.0.0.0:3000
  => Ctrl-C to shutdown server; call with --help for options
  [2006-03-15 13:48:03] INFO WEBrick 1.3.1
  [2006-03-15 13:48:03] INFO ruby 1.8.4 (2005-12-24) [i486-linux]
  [2006-03-15 13:48:03] INFO WEBrick::HTTPServer#start: pid=22170 port=3000
  ```

  Si nos colocamos en http://localhost:3000/ debemos ver la pantalla de bienvenida.

En este punto tenemos montada la aplicación para probar contra un servidor WEBrick. Ahora deberíamos copiar todas las clases que tenemos en nuestra carpeta "app" a la carpeta "app" del proyecto creado y hacerl lo mismo con todos los documentos de la carpeta "Config" que nos interesen.

Si quisiéramos que la aplicación funcionase contra un servidor Apache en lugar de WEBrick deberemos seguir el siguiente procedimiento:

- Apagamos el WEBrick:

  $ctrl-c

- Nos aseguramos de haberlo apagado:

  $ps –U www-data

  No debe haber ningún proceso con este nombre.

- Instalamos los módulos de Apache:

  $sudo apt-get install apache2 libapache2-mod-fastcgi

- Creamos un archivo para que Apache2 ejecute RoR:

  $sudo mcedit /etc/apache2/sites-available/ruby

```
<Virtualhost *:8080>
        ServerAdmin admin@it.uc3m.es
        ServerName *
        DocumentRoot /var/www/ruby/public/
        Options Indexes FollowSymLinks MultiViews
        ErrorLog /var/www/apache2_ruby_error.log

           <Directory /var/www/ruby/public/>
              Options ExecCGI FollowSymLinks
              AddHandler cgi-script .cgi
              AllowOverride all
              Allow from all
              Order allow,deny
           </Directory>
</Virtualhost>
```

- Ligamos:

  $ln -s /etc/apache2/sites-available/ruby /etc/apache2/sites-enabled/ruby

- Creamos una instacia de RoR:

  $cd /var/www && sudo rails ruby

- Reiniciamos Apache:

  $sudo /etc/init.d/apache2 restart

  Si nos colocamos en http://localhost:8080/ debemos ver otra vez la pantalla de bienvenida.